Operating Characteristics for the Design and Optimisation of Classification Systems

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.dr.ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op 19 Desember 2007 om 12:30 uur
door Thomas Christopher Wolfgang LANDGREBE
elektrotechnisch ingenieur (Zuid Afrika, University of the Witwatersrand)
geboren te Roodepoort, Zuid Afrika.

Dit proefschrift is goedgekeurd door de promotoren:
Prof.dr.ir. M.J.T. Reinders

Toegevoegd promotor:
Dr.ir. R.P.W. Duin

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof.dr.ir. M.J.T. Reinders,	Technische Universiteit Delft, promotor
Dr.ir. R.P.W. Duin,	Technische Universiteit Delft, toegevoegd promotor
Prof.dr. A.K. Jain,	Michigan State University, USA
Prof.dr. C. Jonker,	Technische Universiteit Delft
Prof.dr.ir. F. Tortorella,	University of Cassino, Italy
Prof.dr.ir. L.J. van Vliet,	Technische Universiteit Delft
Dr.ir. R. Veldhuis,	Universiteit van Twente

ISBN XX-XXXXXXX-X

Operating Characteristics for the Design and Optimisation of Classification Systems

Contents

Foreword

Many scientific and industrial problems require different concepts or classes to be distinguished. These problems encompass applications such as recognition, detection, discrimination and estimation. Examples of the various classes are typically described by some form of measurement, assumed to be indicative of it's class identity. Repeated measurements may reveal inherent patterns/structures that have the potential to be modelled or "learnt" from the data. The goal is to extract sufficient discriminatory information to assign an object reliably.

In some cases an explicit mathematical model can be used to perform the discrimination task. However in many practical applications, measurements corresponding to objects from the same class have an inherent variability. It is also common for data distributions of the measurements corresponding to the various classes to overlap to a degree. Such problems are best posed in a statistical framework, providing a mechanism for handling both the inherent variabilities and the class overlaps. Statistical pattern recognition is concerned with these types of problems, resulting in decision boundaries in "measurement space" that account for both the inter- and intra-class variabilities/distributions. The approach taken typically uses example objects from each class to estimate the nature of the variability and class overlaps. This scientific area has developed a theoretical foundation for these types of problems, tackling fundamental issues such as generalisation, over-training, dimensionality reduction, and coping with small sample sizes. The pattern recognition approach has been applied successfully to a very diverse number of applications, and is finding more applications at an accelerated pace. This in turn introduces new challenges and opportunities for further research.

In this thesis, a specific area of statistical pattern recognition is tackled, concerned with how trained classifiers behave in various types of environments, and how they can be optimised to suit various types of operating conditions. This area is studied from the basis of what is called classifier "operating characteristics". Testing of a trained classification system with a representative test set reveals the performance of a classifier, indicating how well classes are separated, and the degree of error that occurs between classes.

Interestingly, the trained classifier performance can be modified by weighting of the classifier outputs. Any perturbation of these weighting parameters results in a new performance configuration, or operating point. This type of parameter variation can conceptually be seen as varying decision boundaries in feature space. The decision boundaries inherently trade-off the various possible misclassification rates. Thus understanding how different performance configurations relate to these classifier weightings is important in order to optimise a problem (using for example misallocation costs). Operating characteristics provide this level of understanding by characterising the performance that can be achieved for all possible weighting parameter configurations.

A related topic is the fact that the performance of trained classification systems is affected by variations in operating conditions. For example, if prior probabilities vary with respect to those assumed in the training phase, performance of the system will vary. This imprecise knowledge of the operating conditions is inherent to many problems in pattern recognition, leading to unexpected performances, which complicates the design of the classifier. Operating characteristics are very useful for these imprecise problems, because the variation in performance due to new conditions is in fact characterised by the operating characteristic. Now a trained classifier can be evaluated for a range of conditions, helping to decide on a classifier that is suitable over the expected range of conditions.

This thesis consists of a collection of published conference and journal papers in the area of classifier operating characteristics. This emerging research area is proving to be very useful and important for pattern recognition. The works presented consider the use of operating characteristics in a wide variety of scenarios, contributing both to standard practices, and also considering new areas. First contributions are made to the more traditional 2-class operating characteristic. Next the generalisation of some 2-class approaches to the more elusive multiclass case is considered. Even though this is shown to be theoretically possible, the computational complexity is restrictive for all problems barring those with few classes. The thesis presents a number of approaches to deal with this limitation, with the most important finding being that multiclass operating characteristics can often be simplified considerably due to a lower inherent complexity. It is also shown that operating characteristics are useful tools for designing classifiers in the ill-defined domain, where all classes may not be sampled representatively. Finally multi-stage classifier systems are considered, arguing that constructing an overall operating characteristic for the system is the optimal approach. The works in this thesis are applicable to many new challenges that are emerging in pattern recognition. It is anticipated that this may form the basis for further research.

Chapter 1

Introduction

1.1 Background

The field of Statistical Pattern Recognition (SPR) is concerned with how to go about separating/discriminating various types of concepts/classes. Such problems are common in many scientific and industrial problems for tasks such as data analysis, identity recognition, and object sorting. Discrimination decisions are made based on measurements taken from objects originating from these classes. As the name suggests, SPR is targeted at applications that involve data from which structures/patterns can be modelled/learnt, with the objective of assigning new unseen objects reliably. For example, in hand-written digit recognition, it is of interest to distinguish between various types of digits, typically via images of particular symbols.

The mechanism for separating the data into classes is called a "classifier". A classifier consists of a mathematical construct that assigns objects to various classes, depending on where in "measurement space" they occur. The nature of the partitioning of this space is dependent on the design/architecture of the classifier. In few cases, a more analytic approach can be taken to solve the problem via known data distributions, but in the vast majority of cases, a classifier is constructed based on representative example objects from each class. Representivity is important since SPR applications generally involve statistical variability, which should be discovered so that the domain and distribution of the data can be inferred, as well as the occurrence prevalence (prior probability). Importantly (for this thesis), the trained classifier partitions the measurement space corresponding the the various classes, but these partition boundaries (decision thresholds) can be modified at will to adjust the classifier performance. This becomes important when class overlap exists, and trade-offs need to be made.

The types of problems fitting into the SPR domain have generally presented a number of common core challenges, which have been a primary focus of SPR in the past. Several good texts are available in the field, which study these in detail, e.g. [3], [4], [1]. Some of the most important concepts are briefly mentioned as follows (which are somewhat inter-related):

- **Curse of dimensionality/small sample sizes** SPR problems are often high dimensional, i.e. measurements contain a multitude of dimensions (features), for example images, spectra, signals etc. In the practical domain, there is frequently the dilemma that the number of training samples available does not track the dimensionality (cost/computational limitations). Thus sufficient samples do not exist to estimate the large number of parameters required to train classifiers in this high dimensional space. Fortunately measurements in SPR problems frequently contain redundant information (e.g. noisy/correlated features) that imply that a lower "intrinsic dimensionality" exists. The topic of feature reduction is concerned with reducing the dimensionality of the problem, often resulting in a more tractable situation.

- **Classifier generalisation** A classifier model may be found to fit well to training data, but this performance may not generalise appropriately to new unseen data due to overfitting on the training set (assuming all data originates from the same data distribution). Careful attention must be taken to estimate how well a classifier is expected to generalise, typically by splitting given data into independent sets so that only a portion is used for training, and the remainder for testing e.g. cross- validation.

- **Classifier complexity** Theoretically the classifier should be chosen such that data is separated optimally e.g. by fitting a model to the data distribution. Two confounding factors are prevalent: firstly the fact that data is often high dimensional, it is not trivial to assess whether a "good fit" has been achieved; and secondly, in some cases training data is severely limited, restricting the classifier complexity since estimating a growing number of parameters requires larger training sets. In the former case, classifier performance is indirectly measured using evaluation approaches such as cross-validation. In the latter case, the use of "learning curves" has found to be useful, plotting classifier performance as the training set size increases. SPR has found that adjusting the classifier complexity to suit both the data distribution and the size of the training set leads to good generalisation.

This brief introduction to SPR only intended to give a very brief look at where this field is active, and the types of problems that are involved. Next the specific thesis focus area is discussed, focusing on some specific aspects of the classifier design process.

1.2 Introducing operating characteristics

Operating characteristics pertain to trained SPR systems. This implies that internal classifier parameters have been fixed, and the system is ready to deploy. The trained classifier attempts to discriminate between various classes, partitioning the inherent feature space into regions corresponding to each class. In many problems, there are overlaps between the various classes, and thus the partitioning is not ideal. A fundamental design question is how to go about optimising the partitioning, and trade-off the various classification errors in the most optimal fashion according to the problem requirements. A related question arising in SPR is what impact new operating conditions (e.g. varying class abundances) have on classifier performance.

A trained classifier can be evaluated via a representative test set, resulting in a confusion matrix which demonstrates both intra- and inter-class performance. Thus the error-rate between various classifier outcomes is quantified, as well as the accuracy per class. Importantly, this evaluation reveals merely one possible "performance configuration" that the classifier is capable of. In fact, the classifier can present many different performance outcomes. An explicit way in which to vary the classifier performance is to weight the C classifier outputs (for a $C-$class problem). For example, consider the 3-class synthetic problem illustrated in Figure 1.1. A Bayes quadratic classifier has been trained on this problem for a balanced prior probability scenario, resulting in the decision boundary depicted by the solid line. Weighting of the classifier outputs by [1.0 0.3 0.8] results in the dotted line. This illustrates how the same trained classifier can be manipulated to vary it's performance configuration.

It is easy to see that the performance of a trained classifier can be modified by weighting of the classifier outputs. The important question is how the weighting is related to the resultant performance configuration. Such information is necessary to optimise a classifier to a particular problem e.g. some errors may be more costly than others. Operating characteristics are the mechanism for relating performances to classifier output weightings. Thus all possible performance configurations are defined by the operating characteristic, allowing for the interactions between the various classifier outputs to be evaluated. As such, they provide a convenient mechanism for optimising a trained classifier to suit conditions e.g. to known misallocation costs. A new operating point can be seen to be a movement in the space of the operating characteristic. This "space" is conceptually an evaluation space, with no relation to the feature space.

In a similar way to the manner in which a trained classifier's performance is varied by weighting the classifier outputs, a variation in operating conditions, (a variation in class abundances), results in a different performance configuration. This type of situation occurs in a number of problems, where for example

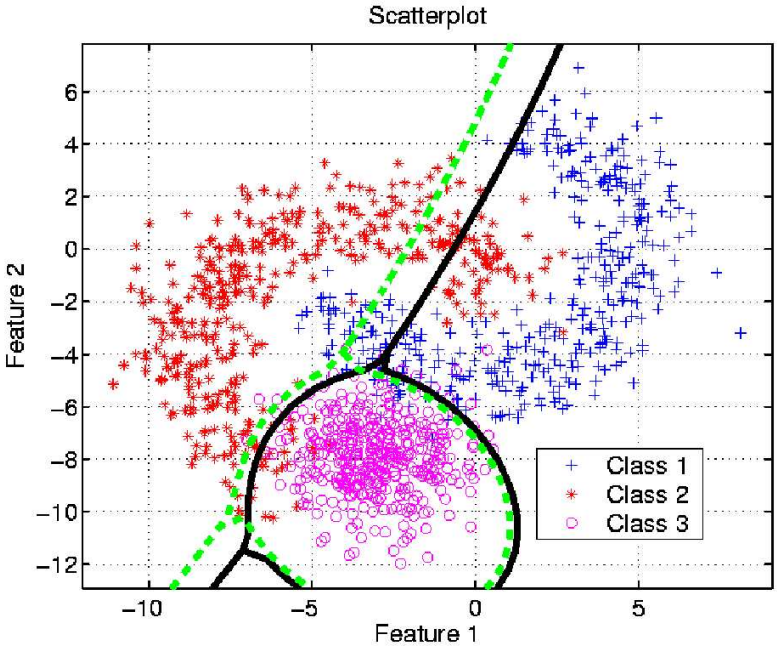


Figure 1.1: Three-class problem, illustrating a Bayes quadratic classifier at two different operating points (solid and dotted lines respectively).

the class abundances assumed in the training phase are different in reality (called imprecise environments [19]). Such a situation is problematic, since a different performance configuration may not be sufficiently accurate. Operating characteristics are very useful in this domain, since all possible performance configurations are characterised. Thus they are a very useful tool for assessing how well a classifier performs in new situations, helping to choose models that cope best with the imprecision.

The works in this thesis demonstrate the importance of operating characteristics for pattern recognition in a wide variety of situations, arguing that they should be an integral part of designing a pattern recognition system. Consider the block diagram in Figure 1.2. The diagram depicts a typical design chain that is used in pattern recognition, and the role that operating characteristics play[1]. In the first step, the problem specification and behaviour are analysed, and an appropriate evaluation designed. The evaluation should indicate if a

[1]Note that this design chain is oversimplified for brevity. A more realistic depiction would involve feedback between the various stages. This is usually necessary as more information about a problem becomes available, and promising initial results justify more investment into the problem (e.g. providing resources for further data capture).

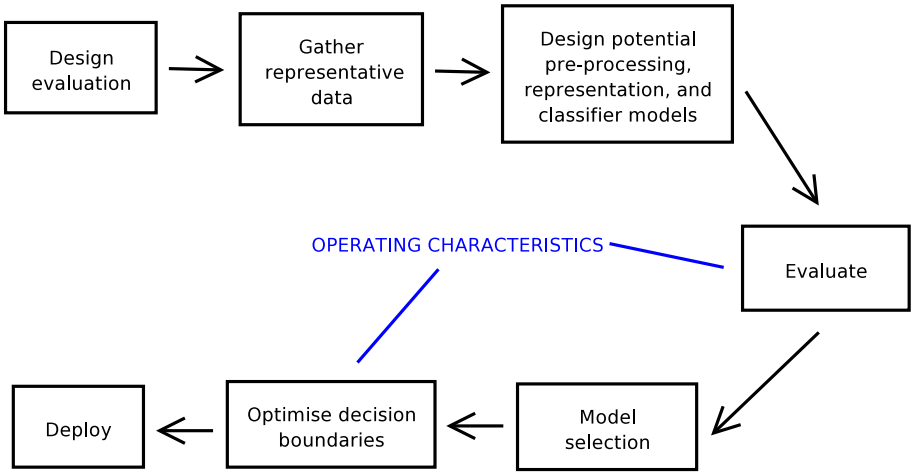


Figure 1.2: Typical pattern recognition system design chain, illustrating where operating characteristics play a role.

particular classifier system will meet the necessary specifications, and it should also consider aspects such as varying operating conditions, and even varying system requirements. The next step is to gather representative data (actual distributions are seldom known) that can be used to train and test classifiers that are suited to the problem. Note that in pattern recognition, obtaining large enough datasets is notoriously difficult and expensive, leading to several core research areas such as coping with small sample sizes the curse of dimensionality, dimensionality reduction and classifier complexity [3], [1], [4]. The next step in the design chain is the classifier design, involving pre-processing, representation, and the classifier model. A typical practice for complex problems is to investigate a number of different models, and even to combine models [6] in order to capitalise on inter-model diversity. The various candidate classifiers are then evaluated, guiding towards the best classifier choice. This step often involves the use of operating characteristics to evaluate performance over multiple operating points (specific setting of decision thresholds), or a range of operating conditions, or even a range of misallocation costs[2]. Subsequent to model selection, the classifier can be prepared for deployment by optimising the decision thresholds to best suit operating conditions. This is another part of the design chain at which operating characteristics are particularly useful. They can be used directly to obtain an equal-error or Bayes operating point, or to select an operating point to suit misallocation costs and priors [1]. Alternatively, a Neyman-Pearson type optimisation can also be performed [3], in which one classifier error is fixed, and the dependent ones minimised. Fi-

[2]Designing classifiers for these imprecise environments has become extremely important in pattern recognition because many practitioners have realised that assuming conditions remain very well-defined is often unrealistic.

nally, once the selected classifier has been trained and optimised, it is ready for deployment.

The description of the classifier system design chain presents a satisfying strategy for going about designing and optimising classification systems. The works presented in this thesis have contributed towards this design philosophy by addressing some gaps, and generalising various strategies to a wide variety of scenarios. This thesis also generalises operating characteristics to some new areas in pattern recognition. In particular, these involve operating characteristics to aid in classifier design in the ill-defined domain [2] (some classes are poorly represented), and also in the optimisation of multi-stage classifier systems. Each chapter is dedicated to one particular aspect of operating characteristic analysis. The various chapters start with a short overview, motivating the research, discussing some research outcomes, and pointing out some open challenges that require further research. These are followed by contributions to each area in the form of published conference and journal papers.

1.3 Outline

In Chapter 2 the traditional 2-class Receiver Operator Characteristic is considered. Two contributions to this area are presented. The first considers the well-known Area Under the ROC (AUC) measure, that evaluates a classifier independently of operating point and operating conditions by integrating over the ROC. This was published in [9]. The emphasis for this contribution is on the application in which prior probabilities vary. It is well known that performance fluctuates along the ROC as the conditions vary. It is shown that when comparing classifiers in this domain, it is important to consider both the integrated classification performance (AUC), and the sensitivity to the expected variation in conditions. For example, two classifiers may have a competing AUC, but one may be less sensitive to a variation in conditions, which may be preferable. The second part of Chapter 2 considers the topic of precision-recall analysis, which are popular evaluation criteria for problems where there is a significant class imbalance/skew, or rare-event problems. This work has been published in [7], based on previous works in [15]. It is shown that precision-recall operating characteristics can be derived directly from the ROC. These new operating characteristics (called P-ROC curves) vary as a function of a variation in class skew in the test/application phase, resulting in a 3-dimensional evaluation surface. A methodology for designing classifiers in the imprecise domain is presented, involving the development of new performance criteria that integrate across the operating surface.

In Chapter 3, the extension of some well-known 2-class ROC analyses are considered for the multiclass case. Whereas ROC analysis is well understood and applied in the 2-class case, generalising this to the multiclass case has not

received much attention due to a number of challenges. The multiclass extension is important for pattern recognition because it broadens the scope to many new potential applications. The first part of Chapter 3 considers the extension of the Neyman-Pearson optimisation strategy to the multiclass case, which was presented in [8]. This strategy involves the specification of a particular classification outcome (as defined by the confusion matrix), with minimisation of the complementing outcome in the 2-class case. The multiclass extension considers the situation in which there are many different classifier outcomes. A practical algorithm is presented that allows one or more outcomes to be fixed, with the remaining outcomes optimised by interrogating the multiclass ROC. A solution is only guaranteed if one outcome is specified. The second part of Chapter 3 presents a simplified extension of the AUC to the multiclass case, resulting in a simplified Volume Under the ROC hyperSurface (VUS). This work has been published in [13], based on [10]. The approach involves generating a $C-$dimensional operating characteristic for a $C-$class problem, followed by a numerical integration procedure to estimate the volume accurately. An important part of the research is consideration of the performance bounds between a perfect and random classifier (e.g. 1 and $\frac{1}{2}$ respectively in the AUC case), since these vary with C. The research shows that the lower bound is simply $\frac{1}{C!}$. Cost-sensitive optimisation is also pertinent to this Chapter, but the extension to the multiclass case is trivial once the multiclass ROC has been generated.

Chapter 4 is concerned with multiclass ROC analysis for large numbers of classes. The primary restriction for ROC analyses such as those discussed in Chapter 3, is that the computational complexity of the ROC calculation increases exponentially as the number of classes increases. These computational considerations are discussed in the first part of Chapter 4, which has been published in [11]. Also presented are a number of algorithms that are designed to perform multi-class cost-sensitive optimisation in an efficient manner. In particular, a pairwise algorithm is formulated that optimises classifier decision thresholds/weights by using the most appropriate 2-class ROC curves that are generated between all class pairs. The second part of Chapter 4 presents an approach that efficiently approximates the full multiclass ROC by decomposing the problem according to interactions between classes. This contribution has been published in [12]. The decomposition results in various classes being treated either independently or in groups, which may in many cases transform an intractable calculation into a tractable one. The justification for this philosophy is based on observations made in many real problems, which revealed that many ROC dimensions are often (approximately) independent. An algorithm is presented that efficiently analyses the interaction between ROC dimensions based on interpreting classifier weight perturbations via the confusion matrix. This results in a unified multiclass ROC approach that can be used directly to perform any type of ROC analysis, extending to large numbers of classes.

In Chapter 5, operating characteristics are used to design, optimise, and evaluate classifiers in ill-defined environments. In these problems, one class is typically well defined, and another is poorly defined, or new unseen classes/clusters could occur during the application phase. For example in road sign recognition [18], the various road-sign classes can be modelled representatively, but the distribution of non-signs that occur in images cannot be modelled. The design objective in these problems is typically to obtain good discrimination performance between known "target" classes, and to protect these known classes from new unseen conditions i.e. new objects that do not originate from the "target" classes should be rejected. The first part of Chapter 5 presents a new rejection scheme that investigates the combination of a standard supervised classifier trained between known classes, with a rejection stage that protects the known classes from "non-target" classes. This work was presented in [17]. This two-stage approach uses a one-class classifier [20] to perform the rejection. It is shown that using different representations and models for the tasks of classification and rejection respectively is often beneficial. This allows each stage to be designed according to it's objective. The second part of Chapter 5 considers the fact that increasing the degree of protection against unknown conditions decreases the classification performance. This was published in [16]. The inherent trade-off/interaction is investigated via a 3-dimensional operating characteristic that accounts for all combinations of the classification and rejection thresholds respectively. Such a framework is useful in selecting both the best rejector-classifier combination, and for choosing the most appropriate thresholds.

In Chapter 6, the use of operating characteristics for multi-stage classifiers is considered, specifically focused on 2-stage recognition systems that use a detector in the first stage, followed by a classifier in the second. This contribution was published in [14]. An operating characteristic is constructed by considering the variation of thresholds in both stages. This allows the entire system to be optimised holistically, accounting for both inter-class and inter-stage interactions.

This thesis by no means exhausts the possibilities and potential of operating characteristics for pattern recognition. It is anticipated that this area will receive much attention in the future. The thesis is not ordered chronologically, but rather in a preferred reading order, starting with classical 2-class ROC analysis, moving to multiclass ROC analysis, and then to ROC analysis applied to ill-defined conditions and multi-stage systems.

1.4 Future perspectives

Tracking academic and industrial trends, it is quite clear that the field of pattern recognition is extending and diversifying. Traditional pattern recognition

problems in which design and evaluation was often simple, and conditions remained stable, are now being replaced by those in which design and evaluation is frequently complex, and conditions less stable. Data is also increasing in dimensionality, redundancy is ever-present, and new sensors are rapidly becoming available, to name a few progressions. The consequence of these trends is that new approaches and techniques are required, some of which have been proposed in this thesis. In anticipation of future challenges and opportunities for pattern recognition, it is useful to consider a few noteworthy trends that have been observed.

1.4.1 Towards problems with very large numbers of classes

Several emerging pattern recognition problems involve a very large numbers of classes C. Examples include speech recognition and remote- sensing terrain mapping [5]. A number of challenges must be faced, such as coping with ill-defined and imprecise environments, as well as optimising operating points to suit conditions. A factor that becomes increasingly dominant with increasing C is the escalation of computational complexity of standard approaches. For example, training a classifier, or constructing a multiclass operating characteristic may be severely limited.

Another consequence inherent to this challenge stems from a practitioner's perspective. As C increases it becomes more difficult to assess performance, and supervise the design in detail throughout the problem. Since the confusion matrix has C^2 outputs, even a 10-class problem has 100 different outputs, and thus careful inspection of the system's performance becomes elusive.

It is clear that more tools and philosophies are required to face problems with large numbers of classes. Even though Chapter 4 considered some of these, many open areas exist, which is an exciting opportunity for future research.

1.4.2 Cheap, mass-produced sensors

It is very apparent that there is currently a strong drive to mass-produce sensors that were once out of reach of many industrial applications (cost/complexity). Two stimuli are attributed to this trend. The first is because it is often the case that the precision offered by some traditional sensors is not necessary, and a simpler, cheaper variant is acceptable. The second stimulus is attributed to efficient manufacturing processes and economies of scale.

The impact on pattern recognition is already evident (for example the areas of hyperspectral imaging, x-ray imaging, and video-based surveillance), but this is expected to increase even more radically with time. Challenges posed will include coping with extreme data redundancy, fusion of multitudes of sensors to leverage complementary information, and dealing with poor/ineffective sensors that constitute a portion of the system.

1.4.3 Increasing computing power

A major challenge in the pattern recognition field is computational complexity, which poses severe limitations. Examples include training of support vector classifiers on large datasets, and construction of the multiclass ROC. However, as computational power increases rapidly, this provides new opportunities for designing better classifier systems. Now it becomes feasible to design more complex classifiers, and to optimise larger systems. Related to this is increasing parallelisation, which has a similar impact.

1.4.4 Holistic design

The core of traditional pattern recognition has typically focused on the design and optimisation of the classifier as part of a greater system. It is, however, becoming apparent that considering the entire system can be beneficial for both evaluation and optimisation. For example, integrating pre-processing, feature extraction, and classification should yield more optimal systems than designing each stage independently (Chapter 6 demonstrates this on an application involving recognition systems). Pattern recognition certainly has a lot of scope in this area, possibly taking a fresh look at traditional problems. A challenge involved is ensuring that the design of each component/stage considers the entire system performance as an evaluation criterion.

1.4.5 A mind-shift - from analytics to inference

It could be argued that in some fields such as signal processing, chemometrics, and even physics, the typical approach that is taken is an accurate analytical one. For example, in materials discrimination using spectroscopy, two material types could be separated using information based on spectral peaks related to the underlying chemistry. However it is becoming increasingly renowned that data-driven approaches such as those provided by statistical pattern recognition can often perform such tasks in a far simpler manner, and make use of apparently ineffective features. In this case class membership is inferred based on various features/measurements. These are also very convenient in accounting for inherent intra- and inter-class variances, and class overlaps. In some applications such as face recognition, shape discrimination, and texture-based classification, it is also difficult to formalise a manual classification procedure, whereas a data-driven approach is often very effective. This circumvents the necessity of imposing a potentially cumbersome model.

Bibliography

[1] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press Inc., New York, first edition, 1995.

[2] B. Dubuisson and M. Masson. A statistical decision rule with incomplete knowledge about classes. *Pattern Recognition*, 26(1):155–165, 1993.

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

[4] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2nd edition, 1990.

[5] S. Kumar, J. Ghosh, and M. M. Crawford. Hierarchical fusion of multiple classifiers for hyperspectral data analysis. *Pattern Analysis and Applications*, 5(2):210–220, 2002.

[6] L.I. Kuncheva and C.A. Whitaker. Measures of diversity in classifier ensembles. *Machine Learning*, 51:181–207, 2003.

[7] T.C.W. Landgrebe, A.P. Bradley, P. Paclík, and R.P.W. Duin. Precision-Recall operating characteristic (P-ROC) curves in imprecise environments. *18th International Conference on Pattern Recognition (ICPR 2006), Hong Kong, China*, August 2006.

[8] T.C.W. Landgrebe and R.P.W Duin. On Neyman-Pearson optimisation for multiclass classifiers. *Sixteenth Annual Symposium of the Pattern Recognition Assoc. of South Africa*, November 2005.

[9] T.C.W. Landgrebe and R.P.W Duin. Combining accuracy and prior sensitivity for classifier design under prior uncertainty. *Structural and Syntactic Pattern Recognition, Proc. SSPR2006 (Hong Kong, China), Lecture notes in computer science vol. 4109, Springer Verlag, Berlin*, pages 512–521, August 2006.

[10] T.C.W. Landgrebe and R.P.W. Duin. A simplified extension of the area under the ROC to the multiclass domain. *Seventeenth Annual Symposium of the Pattern Recognition Association of South Africa*, November 2006.

[11] T.C.W. Landgrebe and R.P.W. Duin. Approximating the multiclass ROC by pairwise analysis. *Pattern Recognition Letters*, 28(13):1747–1758, 2007.

[12] T.C.W. Landgrebe and R.P.W. Duin. Efficient multiclass ROC approximation by decomposition via confusion matrix perturbation analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (accepted)*, 2007.

[13] T.C.W. Landgrebe and R.P.W Duin. A simplified volume under the ROC hypersurface. *Transactions of the South African Institute of Electrical Engineers (accepted)*, 2007.

[14] T.C.W. Landgrebe, P. Paclík, D.M.J. Tax, and R.P.W. Duin. Optimising two-stage recognition systems. *International workshop on multiple classifier systems*, June 2005.

[15] T.C.W. Landgrebe, P. Paclík, D.M.J. Tax, S Verzakov, and R.P.W Duin. Cost- based classifier evaluation for imbalanced problems. *Proceedings of the 10th International Workshop on Structural and Syntactic Pattern Recognition and 5th International Workshop on Statistical Techniques in Pattern Recognition, Lisbon, Portugal*, pages 762–770, August 2004.

[16] T.C.W. Landgrebe, D.M.J. Tax, P. Paclík, and R.P.W. Duin. The interaction between classification and reject performance for distance-based reject-option classifiers. *Pattern Recognition Letters, Special issue on ROC analysis*, 27(8):908–917, June 2006.

[17] T.C.W. Landgrebe, D.M.J. Tax, P. Paclík, R.P.W Duin, and C.M. Andrew. A combining strategy for ill-defined problems. *Fifteenth Annual Symposium of the Pattern Recognition Association of South Africa*, pages 57–62, November 2004.

[18] P. Paclík. Building road sign classifiers. *PhD thesis, CTU Prague, Czech Republic*, December 2004.

[19] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42:203–231, 2001.

[20] D.M.J. Tax. One-class classification. *PhD thesis, Delft Technical University, The Netherlands*, June 2001.

Chapter 2

Two-class operating characteristics

2.1 Overview

Two-class operating characteristics have received a lot of attention in the statistical pattern recognition community, commonly known as Receiver Operator Characteristic (ROC) analysis. Though this analysis has been restricted to 2-class problems, these form a large portion of problems in pattern recognition. The bulk of early pattern recognition problems tackled were well-defined, allowing simple evaluations such as classification error-rate to be used. As pattern recognition extended into new application areas, several challenges were encountered, such as imprecise knowledge of prior probabilities, varying prior probabilities, imbalanced misallocation costs, and different performance criteria. The ROC emerged as a unified tool to deal with these challenges.

One of the most important outcomes of ROC analysis has been the design of an evaluation criterion derived from the ROC, called the Area Under the ROC (AUC) [2]. The AUC considers an integrated performance across a range of imprecision, or operating points. The ROC also provides a convenient mechanism for inspecting the interaction between the classification errors, guiding sensible trade-off choices. The ROC can thus be used directly for cost-sensitive [1] and Neyman-Pearson [3] optimisation.

In this chapter, two contributions to this area are presented. The first considers classifier evaluation for problems in which prior probabilities vary. The standard approach uses the AUC to evaluate this problem, but this ignores an important aspect, namely the performance sensitivity to a change in conditions. It is shown that in some cases, two classifiers may compete in terms of AUC, but have significantly different sensitivities. Thus the approach suggests combining both the AUC and the performance sensitivity. The second contribution considers the extension of standard ROC analysis to the precision-recall case. This type of evaluation is important for applications such as retrieval and rare event detection, where class-skew desensitises evaluations such as error-rate with respect to minority class performance. It is shown that the derived Precision-Recall Operating Characteristics (P-ROC's) have a 3-dimensional nature, with the third dimension necessary to account for precision variation due to class skew. An analysis is presented that extends AUC concepts in the ROC case to the P-ROC case, allowing for evaluation under imprecise circumstances.

Even though ROC analysis has received quite some attention lately, research is still on-going. In particular, an open question has been how to statistically compare ROC's from different classifiers when there is variability in both types of classification errors. In [6], a thorough study of this has been performed. Another recent contribution has been how to adapt ROC analysis to instance-varying costs, as presented in [4]. It is anticipated that 2-class ROC analysis research will continue in the future, investigating areas such as the extensions to other performance criteria, the use as a classifier optimisation criterion (see e.g. [5]), and the understanding of the impact of poor data representivity.

Bibliography

[1] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press Inc., New York, first edition, 1995.

[2] A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

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

[4] T Fawcett. ROC graphs with instance-varying costs. *Pattern Recognition Letters, Special issue on ROC analysis*, 27:882–891, 2005.

[5] D.M.J. Tax, R.P.W. Duin, and Y. Arzhaeva. Linear model combining by optimizing the area under the ROC curve. *18th International Conference on Pattern Recognition (ICPR 2006), Hong Kong, China*, pages 119–122, August 2006.

[6] J. Tilbury, P.W.J. Van Eetvelt, J.M. Garibaldi, J.S.H. Curnow, and E.C. Ifeachor. Receiver operating characteristic analysis for intelligent medical systems - a new approach for finding confidence intervals. *IEEE Transactions on Biomedical Engineering*, 47(7):952–963, July 2000.

2.2 Combining accuracy and prior sensitivity for classifier design under prior uncertainty

This section has been published as 'Combining accuracy and prior sensitivity for classifier design under prior uncertainty', by T.C.W. Landgrebe and R.P.W. Duin, in *Lecture notes in computer science vol. 4109, Springer Verlag, Berlin, Structural and Syntactic Pattern Recognition, Proc. SSPR2006 (Hong Kong, China)*,512-521, August 2006

Abstract

Considering the classification problem in which class priors or misallocation costs are not known precisely, receiver operator characteristic (ROC) analysis has become a standard tool in pattern recognition for obtaining integrated performance measures to cope with the uncertainty. Similarly, in situations in which priors may vary in application, the ROC can be used to inspect performance over the expected range of variation. In this paper we argue that even though measures such as the area under the ROC (AUC) are useful in obtaining an integrated performance measure independent of the priors, it may also be important to incorporate the *sensitivity* across the expected prior-range. We show that a classifier may result in a good AUC score, but a poor (large) prior sensitivity, which may be undesirable. A methodology is proposed that combines both accuracy and sensitivity, providing a new model selection criterion that is relevant to certain problems. Experiments show that incorporating sensitivity is very important in some realistic scenarios, leading to better model selection in some cases.

2.2.1 Introduction

In pattern recognition, a typical assumption made is that class priors and misallocation costs are known precisely, and hence performance measures such as classification error-rate and classifier loss are typically used in evaluation. A topic that has received a lot of attention recently is the imprecise scenario in which these assumptions do not hold (see for example [9], [2], [1] and [10]), resulting in a number of tools and evaluations suited to this problem. In particular, receiver operator characteristic (ROC) curves [6] have become very popular due to their invariance to both class priors and costs, and are thus used as a basis for performance evaluation and classifier decision threshold optimisation in these imprecise environments. The Area Under the ROC (AUC) measure has thus been proposed, providing a performance evaluation that is independent of priors.

In this paper we argue (and show) that considering the integrated performance (AUC) alone may not be the optimal strategy for model selection in these situations. This is because the AUC measure discounts an important characteristic, namely the performance *sensitivity* across the prior range (we distinguish prior sensitivity from the sensitivity measure often used in medical decision making, which is equivalent to true positive rate). In fact, we show that in some cases, two classifiers may compete in terms of AUC, but have significantly different sensitivities over the same prior range i.e. one of the classifiers may have a performance that varies rapidly from low to high values, whereas the other may be more stable. In some problems e.g. medical decision making, the former scenario may be unacceptable, emphasising the fact that this sensitivity should also be considered. A simple criterion is proposed

that combines both AUC and sensitivity, called $AccSens$, allowing for a more appropriate criterion for some problems[1].

The paper is organised as follows: Section 2.2.2 introduces the notation in the well-defined case, restricted to two-class problems for simplicity, and derives the ROC. In Section 2.2.3, the problem of uncertain/varying class priors is considered, discussing the AUC measure, which is invariant of priors. Section 2.2.4 discusses the importance of considering prior-dependent sensitivity in conjunction with integrated error, illustrated via a case study, and Section 2.2.5 subsequently introduces a new criterion, $AccSens$. A number of real experiments are presented in Section 2.2.6 that show some cases in which competing classifiers (using AUC) have significantly different sensitivities (and vice versa). Conclusions are presented in Section 2.2.7.

2.2.2 Problem formulation and ROC analysis

Consider a 2-class classification task between classes ω_1 and ω_2, with prior probabilities $P(\omega_1)$ and $P(\omega_2)$ respectively, and class-conditional probabilities denoted $p(\mathbf{x}|\omega_1)$ and $p(\mathbf{x}|\omega_2)$. Each object is represented by a feature vector $\mathbf{x}$, with dimensionality d. Figure 2.1 presents an example of a 1-dimensional, two-class example (means at -1.6 and 1.6 respectively, and equal variances of 2), and θ_d represents an equal prior, equal cost operating point.

Two types of of classification errors exist in the two-class case, namely the false positive rate (FP_r), and the false negative rate (FN_r), derived as follows, where θ_w is the classification weight, determining the operating point:

$$\begin{aligned} FP_r(\theta_w) &= (1-\theta_w)P(\omega_2)\int p(\mathbf{x}|\omega_2)I_1(\mathbf{x}|\theta_w)dx \\ I_1(\mathbf{x}|\theta_w) &= \begin{cases} 1 \text{ if } \theta_w P(\omega_1)p(\mathbf{x}|\omega_1) > (1-\theta_w)P(\omega_2)p(\mathbf{x}|\omega_2) \\ 0 \text{ otherwise} \end{cases} \\ FN_r(\theta_w) &= \theta_w P(\omega_1)\int p(\mathbf{x}|\omega_1)I_2(\mathbf{x}|\theta_w)dx \\ I_2(\mathbf{x}|\theta_w) &= \begin{cases} 1 \text{ if } (1-\theta_w)P(\omega_2)p(\mathbf{x}|\omega_2) \geq \theta_w P(\omega_1)p(\mathbf{x}|\omega_1) \\ 0 \text{ otherwise} \end{cases} \end{aligned} \tag{2.1}$$

In the (realistic) case that distributions are not known, but are estimated from data (that is assumed representative), class conditional density estimates are denoted $\hat{p}(\mathbf{x}|\omega_1)$ and $\hat{p}(\mathbf{x}|\omega_2)$, and population prior estimates are denoted π_1 and π_2. These are typically estimated from an independent training set that is assumed drawn representatively from the true distribution. Equation 2.1 can then be extended to this case. The classifier weight θ_w allows for FP_r to be traded off against FN_r (and vice-versa) to suit a given application. A particular

[1]Even though we emphasise a varying/uncertain class prior, the theory and analysis in this paper extends also to the related problem of varying misallocation costs [1], since these both have a similar impact from an ROC perspective in that a variation in either prior or cost results in a varying performance, strictly along the ROC [9]

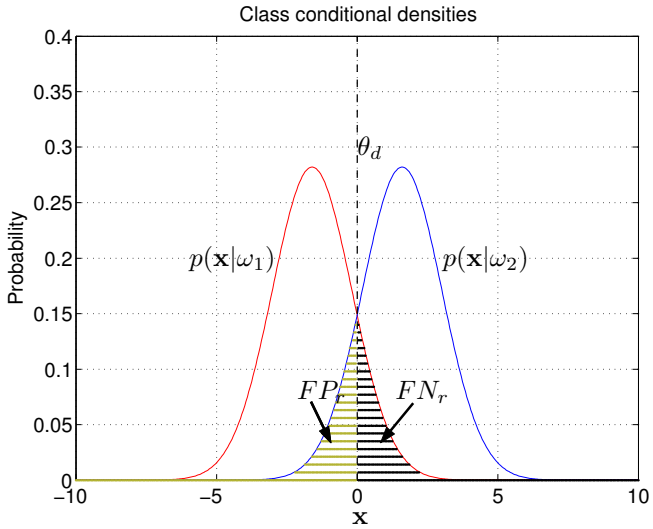


Figure 2.1: One-dimensional example illustrating two overlapping Gaussian distributions, and the two error-types associated with an equal error, equal cost operating point θ_d.

setting of θ_w results in a single operating point, with a corresponding FN_r and FP_r combination. Varying θ_w (where $0 \leq \theta_w \leq 1$) allows for specification of any desired operating point. An ROC plot [6] consists of a trade-off curve between FN_r and FP_r (as a function of θ_w). As such, the ROC is a useful tool in optimising and evaluating classifiers.

In the well-defined case that the priors can be estimated sufficiently well, and remain constant (e.g. estimated from training data, and generalising to an application scenario), the classification problem can be optimised (and evaluated) directly using the ROC. Strategies vary, but the most popular ones are as follows (also demonstrated on the ROC plot in Figure 2.2, which is the ROC plot generated from the example in Figure 2.1):

- **Equal error optimisation**: In this case, FP_r errors have the same consequences as FN_r errors, and the objective of the optimisation is to select a θ_w such that $FP_r = FN_r$. In Figure 2.2, point A shows this operating point.

- **Cost-sensitive optimisation**: In some applications e.g. medical decision making, different errors have different misclassification costs (denoted c_1 for FN_r errors, and c_2 for FP_r errors). In this case θ_w should be chosen such that the overall system loss is minimised, where the loss L can be computed as $L = \theta_w c_1 \pi_1 FN_r + (1 - \theta_w) c_2 \pi_2 FP_r$ (profits are ignored here i.e. consequences of correct classifications). In Figure 2.2, point B

illustrates an operating point for the equal prior case, with $c_1 = 0.2$ and $c_2 = 0.8$.

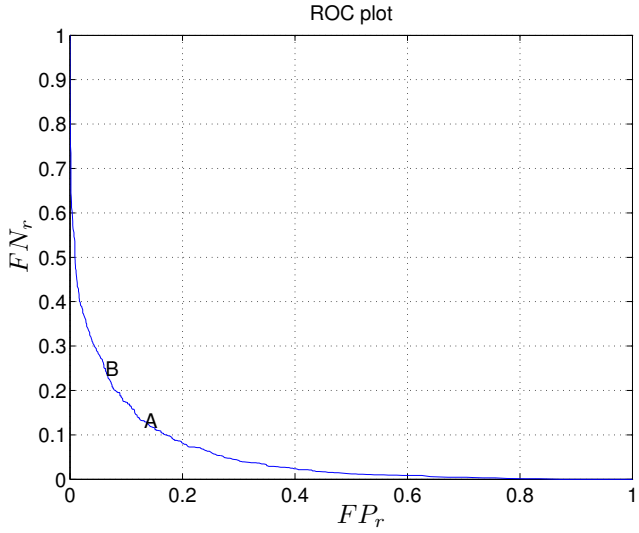


Figure 2.2: ROC plot for the example in Figure 2.1.

2.2.3 Varying priors, uncertain environments

The previous discussion assumed that the priors can be well estimated, and remain fixed in application. However, in many real applications this is not the case (see [9], [2]), confounding the problem of optimising the operating point and model selection (fairly comparing classifiers). In these cases, priors may not be known beforehand, or priors in an independent training set are not representative, or the priors may in fact vary in application. In these cases, even though an immediate optimisation and comparison is not appropriate, several techniques have been proposed for classifier design e.g. [9]. These typically use the ROC plot, since it has the desirable property of being independent of priors/costs (i.e. the same ROC results irrespectively), allowing classifier performance to be inspected for a range of priors (or costs). In particular, the Area Under the ROC (AUC) measure [2] has been derived to give an integrated performance measure, allowing for model comparison independent of the prior. The AUC measure is defined as:

$$AUC = 1 - \int (FN_r) dFP_r \tag{2.2}$$

This performance measure results in a normalised score between 0 and 1, with 1 corresponding to perfect classification, 0.5 to random classification, and below 0.5 as worse than random (i.e. swap classifier labels). The AUC measure

can also be computed over a range of priors/operating points, accounting for knowledge of the degree of uncertainty/variation. Thus, even though priors may be uncertain/varying, the best overall classifier can be chosen based on the most favourable integrated performance[2].

2.2.4 The importance of incorporating sensitivity

In this paper we demonstrate that comparing classifiers in uncertain environments on the basis of integrated error (AUC) only may not necessarily be the best strategy to take. This argument arose based on comparison of ROC plots for a number of competing classifiers (the experiments will show some realistic scenarios). It was observed that in some cases, two competing classifiers resulted in a similar AUC score, but inspection of the ROC made it clear that in one case, the performance range was small, but in another, much larger. This implies that for the problem in which priors may vary, the latter classifier may result in very poor performance at one extreme, and very good performance at the other. Depending on the problem, it may be much better to select the former model that is generally more stable over the expected prior range. Next a case study is presented to demonstrate such a scenario.

Case study

Figure 2.3 depicts a demonstration of a model-selection scenario, comparing two different classifiers, denoted A and B respectively. Each classifier is trained on the distribution shown in the left plot, consisting of a two-class problem between ω_1 and ω_2 respectively, where ω_1 objects are drawn from $N(\mu = 3.0, 2; \omega = 1) + \frac{1}{32}N(\mu = -2.0, 5.0; \sigma = 1)$ (N is the normal distribution with mean μ and variance σ), and ω_2 is one class from the banana distribution [4]. In this synthetic problem, 1500 objects are drawn from the true distribution to create a training set, and a further 1500 objects are drawn independently to result in an independent test set[3]. The two classifiers A and B are then trained on the training set, resulting in the decision boundaries at a single operating point as depicted in the left plot. A is a mixture of Gaussians classifier, with two mixtures chosen for ω_1, and one for ω_2. Classifier B is a support vector classifier with a second order polynomial kernel.

In this problem, it is assumed that the priors may vary (in application) such that $0.05 \leq \pi_1 \leq 0.9$, i.e. the abundance of ω_1 varies between 5% and 90%, and the costs are assumed equal (priors at the low and high extremes for ω_1 are denoted π_1^{lo} and π_1^{hi} respectively, computed by analysing where on the ROC the performance drifts to for the new prior, relative to the original operating point). The scatter-plot shows the resultant classifier decision boundaries of the two classifiers at the equal error point (i.e. equal priors). The ROC plot

[2]For threshold optimisation, the best strategy may be to use a θ_w corresponding to the centre of the known range, or to apply the minimax criterion [3].

[3]Cross-validation is ignored here as this example is for demonstration purposes only.

on the right depicts classifier performance for a range of operating points. For the first extreme, i.e. $\pi_1 = 0.05$, A_{lo} and B_{lo} show the respective operating points for the two classifiers. For the second extreme, i.e. at $\pi_1 = 0.9$, A_{hi} and B_{hi} again demonstrate how the operating point shifts. A_e and B_e show the positions of the equal-error points.

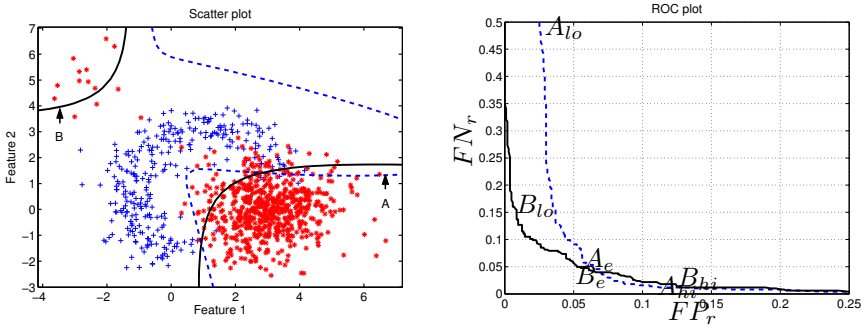


Figure 2.3: Case study illustrating performance of two competing classifier models A and B. The left plot shows the data distribution, as well as the respective decision boundaries at a single operating point. The right plot is an ROC-plot for the two models across a range of priors. A_{lo} and B_{lo} are operating points at $\pi_1 = 0.05$, and similarly A_e and B_e are equal-error points, and A_{hi} and B_{hi} correspond to $\pi_1 = 0.9$.

It can immediately be observed that the two classifiers have a distinct performance characteristic as a function of the prior values, even though the equal error points are rather similar. Table 2.1 compares some performance measures between classifiers A and B. Firstly the error rate shows that both classifiers result in a similar performance for the equal prior case. The AUC measure integrates the classification error over the range of priors (between A_{lo} and A_{hi}), and again this measure shows that both classifiers have similar performance across the prior range as a whole. However, when investigating the sensitivity with respect to the priors, it can be seen that classifier A is much more sensitive than B across the range, with the FN_r varying by up to 47.3%. Prior sensitivity (denoted $Sens$) is computed as the Euclidean distance between the upper and lower prior range, from a π_1^{lo} situation, to π_1^{hi}. This is performed by considering the applicable ranges of FN_r and FP_r:

$$Sens = \frac{1}{\sqrt{2}}\sqrt{((FN_r(\pi_1^{lo}) - FN_r(\pi_1^{hi}))^2 + (FP_r(\pi_1^{hi}) - FP_r(\pi_1^{lo}))^2} \qquad (2.3)$$

This measure scales between 0 and 1, where a low score indicates the favourable condition of low sensitivity, whereas a high score indicates a large sensitivity to prior variation. Note that $Sens$ is a simple measure in that it subtracts only the extreme values, justified by the fact that an ROC increases monotonically.

Model	ϵ	AUC	$Sens$
A	0.057	0.942	0.340
B	0.052	0.945	0.131

Table 2.1: Performance measures for the synthetic example. Error-rate is denoted ϵ, AUC is the integrated error measure across the prior range, and the sensitivity $Sens$ shows how much the performance varies ($\frac{\%}{100}$) across the prior range.

In this type of problem, classifier B is clearly more appropriate since it is far less sensitive to a perturbation in prior. It is also clear that the error-rate measure and AUC are not sufficient on their own in this case to choose the best models, and that the prior *sensitivity* across the range of interest should be included to aid in the model selection process.

2.2.5 Combining accuracy and sensitivity

The case study made it clear that in the uncertain prior situation, classifier sensitivity should be considered in conjunction with integrated error over the prior range. The next step is to develop a criterion that combines these two performance measures, that is useful for evaluation/model selection in this domain. It is conceivable that some problems may have different consequences for accuracy and sensitivity performances e.g. in some cases a low overall error (i.e. high AUC) may be more important than a low sensitivity, in which case $Sens$ could be weighted lower than AUC. In another case, e.g. medical decision making, a high sensitivity to priors may be more unacceptable than a slightly lower AUC. Thus, for generality, we introduce a weighting corresponding to each term, that can be used to penalise either according to the problem (analogous to misallocation costs). The AUC weight is denoted w_e, and the $Sens$ weight is denoted w_s. We then define the combined measure, called $AccSens$, consisting of the geometric mean of the weighted sum of AUC and $Sens$, as defined in Equation 2.4. This is appropriate because both measures are scaled between 0 and 1. In the case that w_e and w_s are both set to unity (equal importance), the $AccSens$ error measure also scales between 0 and 1, where a low score is favourable (the $\frac{1}{\sqrt{2}}$ normalises the measure to this range).

$$AccSens = \frac{1}{\sqrt{2}}\sqrt{w_e((1-AUC)^2) + w_s(Sens^2)} \tag{2.4}$$

For the case study example (assuming unit weighting), the $AccSens$ errors are 0.244 for model A, and 0.100 for model B, indicating that B is superior.

2.2.6 Experiments

A number of experiments on realistic datasets have been undertaken. The objective is to select the most competitive model, considering the problem of varying/uncertain priors, with a known π_1 range: $0.1 \leq \pi_1 \leq 0.9$. Additionally,

we assume AUC and $Sens$ are weighted equally. For each model, we investigate an integrated error over the prior range (AUC), the $Sens$ (sensitivity) across the range (Equation 2.3), the $AccSens$ measure to combine the two, and finally the equal error rate ϵ for comparison purposes. In each experiment, a 10-fold randomised hold-out procedure is performed, effectively resulting in 10 ROC plots upon which the aforementioned statistics are computed. Significance between models is assessed using ANOVA (99.5% significance level). The following datasets are used:

- **Road sign**: A road sign classification dataset [8] consisting of various *sign* and non-*sign* examples represented by images (793 pixels). All *signs* have been grouped together into a single class (381 objects), to be discriminated from non-*signs* (888 objects).
- **Phoneme**: This dataset is sourced from the ELENA project [5], in which the task is to distinguish between oral and nasal sounds, based on five coefficients (harmonics) of cochlear spectra. In this problem, the "nasal" class (3818 objects) is to be discriminated from the "oral" class (1586 objects).
- **Sonar** and **Ionosphere** are two well-known datasets from the UCI machine learning database [7].

Results are presented in Table 2.2. Various representation and classification algorithms have been used. Preprocessing/representation: *sc* denotes unit variance scaling, *pca* is a principle component mapping followed by the number of components used, or the fraction of variance retained, and $fisher$ is a Fisher mapping. Classifiers: *knnc* denotes the k-nearest neighbour classifier followed by the number of neighbours considered, $parzenc$ is a Parzen-window classifier, *ldc* and *qdc* are Bayes linear and quadratic classifiers respectively, *mogc* is a mixture of Gaussians classifier followed by the number of mixtures per class, and *svc* is a support vector classifier, with p denoting a polynomial kernel followed by the order, and r denoting a Gaussian kernel, followed by the variance parameter.

Results show that there are many cases in which incorporation of sensitivity is important for this problem. In the *Road sign* case, an example of this is demonstrated by comparing models 1) and 2). Both show a similar AUC score, but 2) is much less sensitive to prior variation. The $AccSens$ measure is sensitive to this difference, showing significance (based on an ANOVA hypothesis test). Another interesting comparison is between 3) and 4), in which case model 3) has a significantly higher AUC, but 4) has a significantly better $Sens$. Both result in the same $AccSens$ score. Models 3), 4), 5), and 6) all compete from an $AccSens$ perspective (significantly better than 1) and 2)). In the *Phoneme* dataset, model 3) competes with 1) and 2) in terms of AUC, but 2) results in a better $Sens$, and thus results in a superior $AccSens$ score (significant). This clearly illustrates the point of the paper once again - without

Model	AUC	Sens	AccSens	ϵ
Road sign				
1) *pca8 mogc4,4*	0.881(0.026)	0.272(0.039)	0.211(0.029)	0.127(0.022)
2) *pca12 mogc2,2*	0.886(0.058)	0.180(0.029)	0.154(0.028)	0.093(0.021)
3) *sc svc r 16*	0.951(0.016)	0.149(0.028)	0.111(0.021)	0.052(0.014)
4) *pca17 mogc2,4*	0.876(0.100)	0.080(0.026)	0.112(0.056)	0.043(0.017)
5) *sc svc r 22*	0.952(0.016)	0.128(0.019)	0.100(0.015)	0.049(0.013)
6) *pca14 mogc2,4*	0.907(0.061)	0.109(0.021)	0.106(0.033)	0.055(0.016)
Phoneme				
1) *sc knnc3*	0.905(0.013)	0.271(0.049)	0.204(0.028)	0.140(0.011)
2) *sc knnc1*	0.913(0.009)	0.248(0.013)	0.186(0.010)	0.107(0.008)
3) *sc parzenc*	0.891(0.014)	0.294(0.023)	0.222(0.018)	0.128(0.015)
Sonar				
1) *sc knnc3*	0.887(0.027)	0.310(0.107)	0.235(0.073)	0.147(0.039)
2) *sc knnc1*	0.892(0.036)	0.280(0.054)	0.213(0.043)	0.122(0.050)
3) *pca6 parzenc*	0.850(0.050)	0.405(0.069)	0.308(0.046)	0.167(0.054)
4) *sc svc p4*	0.829(0.056)	0.533(0.141)	0.398(0.100)	0.218(0.066)
Ionosphere				
1) *pca0.999 ldc*	0.855(0.039)	0.385(0.118)	0.292(0.084)	0.145(0.043)
2) *fisher qdc*	0.855(0.037)	0.337(0.053)	0.260(0.041)	0.140(0.036)
3) *fisher mogc3,3*	0.834(0.035)	0.365(0.093)	0.285(0.063)	0.160(0.040)
4) *sc svc r 1.0*	0.853(0.171)	0.545(0.231)	0.434(0.095)	0.128(0.044)

Table 2.2: Results of real experiments, comparing AUC, $Sens$, $AccSens$, and ϵ (equal-error point) for a number of models per dataset. Standard deviations are shown.

considering sensitivity, model 3) could have been chosen instead of 1) or 2). In the *Sonar* dataset, model 2) appears superior in terms of both AUC and $Sens$, and thus there was no benefit of the new measure in this case. Finally, in the *Ionosphere* dataset, models 1), 2) and 4) result in similar AUC scores, but 2) appears less sensitive than 4) (not very significant). Using the $AccSens$ measure, 1), 2) and 3) are significantly better than 4). As a final general comment on experimental results, it is apparent that there are cases in which a model selection based on AUC only is not the optimal procedure. Thus, we argue that in the prior uncertain/unstable environment, prior sensitivity should also be considered, using for example the $AccSens$ measure.

2.2.7 Conclusions

In this paper the problem of varying/uncertain priors was investigated. ROC analysis has become a standard tool in this domain, with the Area Under the ROC (AUC) a popular model selection criterion. We argued that even though this integrated measure can be used to compare classifiers independent of priors, it may also be important to consider how *stable* a model is over the

relevant range. A case study and some realistic experiments were presented that demonstrated how classifiers that compete in terms of AUC may differ significantly in terms of sensitivity (and vice-versa). It may thus be more sensible for the given problem to consider both. A simple measure, called $AccSens$ was proposed, that combines the (weighted) geometric means of AUC and sensitivity, allowing for model comparison that considers both integrated accuracy (AUC), and prior sensitivity. A few real experiments demonstrated that this methodology is superior in some situations. **Acknowledgements** This research is/was supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs.

Bibliography

[1] N.M. Adams and D.J. Hand. Comparing classifiers when misallocation costs are uncertain. *Pattern Recognition*, 32(7):1139–1147, 1999.

[2] A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

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

[4] R.P.W. Duin. *PRTools, A Matlab Toolbox for Pattern Recognition*. Pattern Recognition Group, TUDelft, January 2000.

[5] ELENA. European ESPRIT 5516 project. *phoneme dataset*, 2004.

[6] C. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, 3(4), 1978.

[7] P.M. Murphy and D.W. Aha. UCI repository of machine learning databases, ftp://ftp.ics.uci.edu/pub/machine-learning-databases. *University of California, Department of Information and Computer Science*, 1992.

[8] P. Paclík. Building road sign classifiers. *PhD thesis, CTU Prague, Czech Republic*, December 2004.

[9] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42:203–231, 2001.

[10] J Yuen. Bayesian approaches to plant disease forecasting. *Plant Health Progress*, November 2003.

2.3 Precision-Recall Operating characteristic (P-ROC) curves in imprecise environments

This section has been published as 'Precision-Recall Operating characteristic (P-ROC) curves in imprecise environments', by T.C.W. Landgrebe, A.P. Bradley, P. Paclík and R.P.W. Duin, in *18th International Conference on Pattern Recognition (ICPR 2006), Hong Kong, China*, August 2006

Abstract

Traditionally, machine learning algorithms have been evaluated in applications where assumptions can be reliably made about class priors and/or misclassification costs. In this paper, we consider the case of imprecise environments, where little may be known about these factors and they may well vary significantly when the system is applied. Specifically, the use of precision-recall analysis is investigated and compared to the more well known performance measures such as error-rate and the receiver operating characteristic (ROC). We argue that while ROC analysis is invariant to variations in class priors, this invariance in fact hides an important factor of the evaluation in imprecise environments. Therefore, we develop a generalised precision-recall analysis methodology in which variation due to prior class probabilities is incorporated into a multi-way analysis of variance (ANOVA). The increased sensitivity and reliability of this approach is demonstrated in a remote sensing application.

2.3.1 Introduction

In pattern recognition, a common evaluation strategy is to consider classification *accuracy* or its complement error-rate. In many empirical evaluations it is common to assume that the natural distribution (prior probabilities) of each class are known and fixed [9]. A further assumption often made is that the respective misclassification costs are known, allowing for the optimal decision threshold to be found [4]. Here, performance measures such as error-rate may be applied to compare different models as appropriate. However, in imprecise environments, misclassification costs can not be specified exactly, and class priors may not be reflected by the sampling, or even worse, the priors may in fact vary. Consequently, optimal threshold selection is ill-defined, and model selection based on a fixed threshold is unsuitable. For example, in remote sensing [8], the prior probability of various topography classes are not known a-priori, and may vary geographically. In such a situation, a performance measure should allow for an assessment that is either independent of these imprecise/ill-defined conditions or incorporates this variation.

Receiver Operator Characteristic (ROC) analysis [9], [10], has become a useful, and well-studied tool for the evaluation of classifiers in this domain. Measures such as the Area under the ROC (AUC) [10] allow for a performance evaluation independent of costs and priors by integrating performance over a range of decision thresholds. This can then be viewed as a performance measure that is integrated over a region of possible operating points.

In this paper we consider the evaluation of two-class classification problems where positive classes are to be distinguished from negative classes. In an imbalanced setting, where the prior probability of the positive class is significantly less than the negative class (the ratio of these being defined as the *skew* or λ), *accuracy* is inadequate as a performance measure since it becomes biased towards the majority class [13]. That is, as the skew increases, *accuracy* tends towards majority class performance, effectively ignoring the recognition

capability with respect to the minority class. In these situations, other performance measures such as *precision* (in conjunction with *recall*) may be more appropriate as they remain sensitive to the performance on each class. Figure 2.4 compares *accuracy* and *precision* as a function of skew for an example (a linear discriminant trained on the Highleyman distribution [5]), illustrating that as the skew increases, *accuracy* tends towards TN_r (majority class performance), effectively ignoring the recognition capability with respect to the minority class.

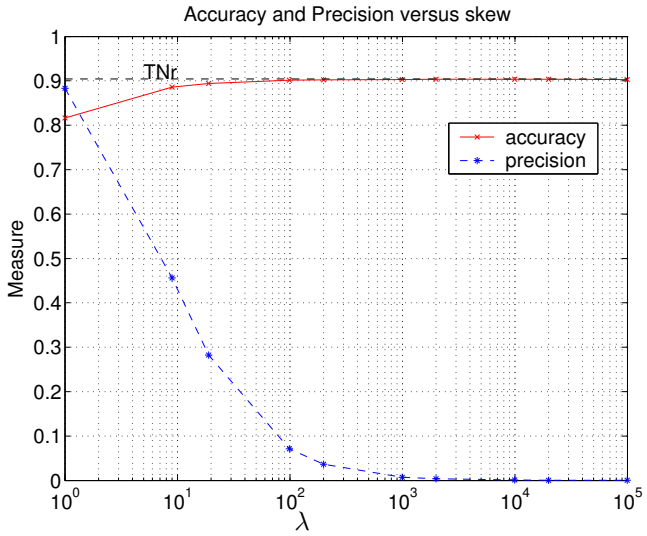


Figure 2.4: Comparing *accuracy* and *precision* for an example, as a function of skew (λ), illustrating the tendency of *accuracy* to approach the majority class performance (TN_r) with increasing skew.

We apply a ROC analysis methodology to the case of *precision-recall* curves. However, we show that because *precision* is dependent upon the degree of skewing, an additional dimension (the *skew*) must be introduced into the analysis. This effectively results in a 3-dimensional ROC *surface*. A similar approach was described in [3], where the relationships between a number of performance evaluation criteria were derived with respect to the ROC curve. In addition, we have previously presented an analysis specific to imbalanced problems, involving *precision* operating characteristics for a number of selected operating points and priors [7]. Here however, we generalise this work so that the evaluation considers the entire operating surface, and integrated performance measures are then derived in a similar way to conventional ROC analysis. The performance of a number of models is statistically compared using a hypothesis-testing framework involving a 3-way analysis of variance (ANOVA) between classification thresholds, priors and models. We demonstrate the approach via a remote sensing application.

2.3.2 Formalisation

Consider a two-class classification problem between a positive and a negative class, ω_p and ω_n respectively, with priors π_p and π_n. An evaluation of a trained model is based on the outcomes following the application of a test set. In the 2-class case this results in a confusion matrix where test objects labelled by the trained classifier as positive fall into two categories: true positives TP and false positives FP. Correspondingly, true positive and false positive rates TP_r and FP_r, are computed by normalising TP and FP by the total number of positive (N_p) and negative (N_n) objects respectively, where N objects are involved in the test ($N = N_p + N_n$). Data samples labelled by the classifier as negative also fall in two categories, true negatives TN and false negatives FN. Also note that $TN_r = 1 - FP_r$, and $FN_r = 1 - TP_r$.

Although a confusion matrix shows all of the information about a classifier's performance, it is usual to extract measures from this matrix to illustrate specific aspects of the performance. For example:

1. Classification *accuracy*, or its complement error-rate (*error*), defined as $error = \frac{FN+FP}{N} = \pi_p FN_r + \pi_n FP_r$. This estimates the overall probability of correctly labelling a test sample, but combines results for both classes in proportion to the class priors;

2. $Recall = TP_r$. This indicates the probability of correctly detecting a positive test sample and is independent of class priors. TP_r is often utilised in medical applications where it is referred to as test *sensitivity*. In medical applications the complement to sensitivity is also used, namely *Specificity* (TN_r). Specificity indicates the probability of correctly detecting a negative test sample and is also invariant of class priors;

3. $Precision = \frac{TP}{TP+FP}$. This indicates the fraction of the positives detected that are actually correct. Precision effectively estimates an overall posterior probability and is therefore a meaningful performance measure when detecting rare events. Precision combines results from both positive and negative samples and so is class prior dependent. It is also often referred to as *purity*, or in medical applications as positive predictive value (PPV). Note: the complement to PPV is negative predictive value (NPV);

4. $Posfrac = \frac{TP+FP}{N}$. This measure is useful in applications requiring second-stage manual processing of the positive outcomes of the classifier (such as medical screening tests), and estimates the reduction in manual effort provided by the classification model.

These measures highlight different aspects of a model's classification performance and so selecting the most appropriate performance measure is clearly application dependent. In medical applications for example, sensitivity (TP_r) and specificity (TN_r) are well understood, can be related to the prior class probabilities, and so are well accepted by the end-users. Therefore, these measures are used almost exclusively in these applications. However, in applications such

as database image retrieval and oil-spill detection from satellite radar images *precision-recall* analysis is more appropriate [6]. In these applications *recall* (TP_r) only really makes sense when combined with *precision*, as the prior class probabilities are unknown or highly variable. In these situations, end-users relate to *precision-recall* curves as they indicate how many true positives are likely to be found in a typical search.

It is also worth noting that in a similar way in which *error* is used as a scalar performance measure in well-defined pattern recognition problems, scalar measures such as the $F-measure$ [11] are used in the well-defined *precision-recall* case (the geometric mean of *precision* and *recall*, in which the two measures are weighted equally), defined as $\frac{2TP_r}{TP_r+FP_r+1}$.

2.3.3 ROC analysis

The performance measures described before all relate to a single decision threshold, or operating point, for a classification model. In well defined environments, where class priors and misclassification costs are known, evaluation at a single (perhaps optimal) operating point is appropriate. However, in imprecise environments or when comparing models operating at different points, ROC analysis is more appropriate.

Given a two class problem (ω_p vs ω_n), a trained density-based classifier and a test set, the ROC curve is computed as follows[4]: the trained classifier is applied to the test set, and the aposteriori probability is estimated for each data sample. Then, a set of m thresholds ($\theta = \theta_1, \theta_2, \ldots, \theta_m$) are applied to this probability estimate and corresponding data labellings are generated. This can be conceptualised as shifting the position of the decision boundary of a classifier across all possibilities. The confusion matrix is computed between each estimated set of labels and the true test-set labelling. The ROC curve now plots the TP_r as a function of the FP_r. This effectively results in a representation of all possible classification *accuracy* values for a given classifier, and provided the train and test data are representative, the same ROC results irrespective of priors/costs.

It is well known that evaluation measures such as *accuracy* vary with prior/cost [10]. Thus a classifier trained to, for example, the Bayes operating point, would report a different *accuracy* as the priors vary. In order to maintain the Bayes error-rate, the decision threshold would have to be adjusted according to the variation in prior/cost. In cases where costs/priors are not defined well, there is a need to inspect performance for a range of different operating points and/or priors. If all operating points are used in the evaluation, the overall ROC curve will be invariant to priors [9]. Integrating performance over the whole ROC curve results in the Area Under the ROC curve (AUC) [1] [10], which is a scalar performance measure ranging from 0.5 (random classification) to 1.0 (ideal). It is also often more practical to compute

[4]The true class-conditional distributions are typically not known, so the method we use to derive the ROC is an estimate of the true ROC.

the AUC over a limited range to suit the given problem.

$$AUC(\theta) \quad = \quad \int TP_r(\theta) dFP_r(\theta) \tag{2.5}$$

This can be approximated non-parametrically via trapezoidal integration:

$$\begin{array}{c} AUC(\theta) \approx \sum_{i=2}^{m} \Delta FP_r TP_r(\theta_i) + \frac{1}{2}\Delta TP_r \Delta FP_r \\ \Delta TP_r = TP_r(\theta_i) - TP_r(\theta_{i-1}) \\ \Delta FP_r = FP_r(\theta_i) - FP_r(\theta_{i-1}) \end{array} \tag{2.6}$$

The point to note here is that while the ROC curve, and therefore AUC, is invariant to priors/costs, in imprecise environments we are actually interested in the variability in performance as the priors vary (we want to select the best performing model across an expected range of priors). Therefore, the traditional ROC analysis tools are not appropriate and require extension to imprecise environments.

2.3.4 Precision-recall analysis

Whereas ROC analysis represents $TP_r(\theta)$ against $FP_r(\theta)$, the *precision-recall* operating characteristics represent $TP_r(\theta)$ against *precision*(θ). As discussed in [7], we showed that *precision* is in fact dependent on the priors, i.e., a new operating characteristic is obtained if the priors vary, as opposed to the ROC where thresholds/operating points and priors are synonymous. The consequence is that the operating characteristic constitutes a surface of operating points, with each prior resulting in a slice of this surface. The *precision* definition can be written as:

$$precision(\theta) = \frac{TP_r(\theta)}{TP_r(\theta) + \lambda FP_r(\theta)} \tag{2.7}$$

This allows the performances to be obtained analytically, given an ROC (derived as in Equation 2.6). In Figure 2.5, an example of receiver (TP_r vs FP_r), and *precision-recall* (TP_r vs *precision*) operating characteristic curves are shown for an example classifier and dataset. The *precision* characteristics are shown for three different prior settings ($\pi_p = 0.5,\ 0.1,$ and 0.01) to demonstrate the prior dependence from a balanced to an imbalanced situation. It is clear that the *precision* characteristic varies significantly with λ.

The AUC is computed by integrating across all classification thresholds θ. Similarly, the *precision-recall* characteristic can be integrated across both classification thresholds θ and priors λ, thus obtaining an integrated performance measure, called $AUPREC$. This can again be derived using the trapezoidal approximation, resulting in Equation 2.8. With this formulation, the original ROC can be used, together with the given skew, to analytically compute the new performance measures.

$$\begin{array}{c} AUPREC(\lambda) = \int TP_r(\theta) dprecision(\theta, \lambda) \\ \approx \frac{1}{2} \sum_{i=2}^{m} \Delta TP_r [\frac{TP_r(\theta_i)}{TP_r(\theta_i) + \lambda FP_r(\theta_i)} + \frac{TP_r(\theta_{i-1})}{TP_r(\theta_{i-1}) + \lambda (FP_r \theta_{i-1})}] \end{array} \tag{2.8}$$

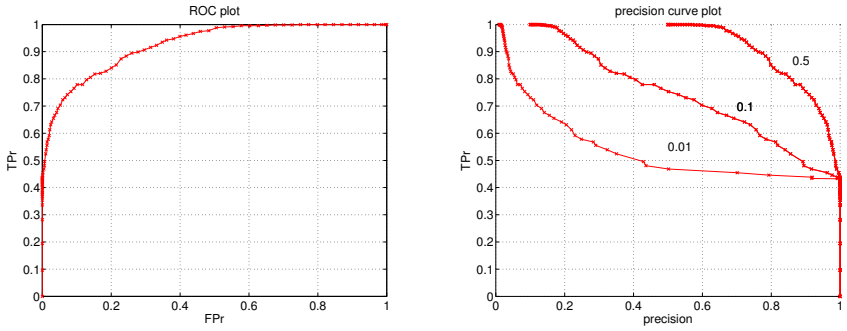


Figure 2.5: Demonstrating an ROC curve (left), and *precision-recall* characteristics (right).

The $AUPREC$ results in a performance score for a single skew setting. However, we wish to estimate performance in problems in which the skew/costs are unknown, or only a range can be specified. In this case we wish to evaluate *precision* across a range of priors. We therefore define an integrated *precision* measure called $IAUPREC$. For a range of skew values (or priors) $\boldsymbol{\lambda} = \{\lambda^{lo}, \lambda^{hi}\}$, we obtain the $IAUPREC$ as shown in Equation 2.9.

$$IAUPREC(\lambda_{lo}, \lambda_{hi}) = \int_{\lambda_{lo}}^{\lambda_{hi}} AUPREC(\boldsymbol{\lambda}) d\boldsymbol{\lambda} \tag{2.9}$$

2.3.5 Hypothesis testing by 3-way ANOVA

In this paper, we use analysis of variance (ANOVA) to test the null hypothesis that a number of models have, on the average, the same performance. If there is evidence to reject this hypothesis then we can look at the alternative hypothesis that one classifier has better performance than the others. ANOVA is simply an extension of Hypothesis tests of means (such as the t and F tests) to the case of multiple groups (in our case, > 2 classifiers) [12]. This avoids the necessity of performing multiple hypothesis tests for each pair of classifiers as we effectively test all hypotheses simultaneously.

ANOVA provides a method for splitting the variation in the data between multiple components (e.g., experimental error, classifier model, cross validation fold and prior probability). If the null hypothesis is true, then all components provide an independent estimate of the experimental error (that is, no components have a significant effect on performance). Clearly we expect that some of these components will affect performance and although we may not interested in them specifically, we use them as *blocking factors* to improve test sensitivity. Conventionally in ANOVA an F-test is used, however other non-parametric tests can also be used (e.g., rank statistics are used in the Friedman test). In this paper, we use a conventional F-test, and we specifically compare the efficacy of a 2-way ANOVA, with IAUPREC as the performance measure, to

a 3-way ANOVA, with AUPREC as the performance measure, and π_p as a blocking factor. All tests are performed at the $p = 0.005$ level of significance, which gives a 1 in 200 probability of rejecting the null hypothesis *by chance*.

2.3.6 Experiments

In this section a number of experiments are undertaken in a real problem domain to demonstrate the efficacy of the proposed *precision-recall* analysis. A remote sensing application is targeted, which we call *Satellite*[5]. As discussed in [8], this problem is appropriate because the prior probabilities of the various classes vary geographically. The data consists of 6435 multi-spectral values of a satellite image, with 36 dimensions (4 spectral bands in a 9 pixel neighbourhood). Six classes have been identified to characterise the topography, of which the second and fourth classes (cotton crop and damp grey soil) are considered ω_p (1329 examples), and the remaining ones ω_n (5106 examples). The goal of the experiments is to select a classifier that remains relatively robust to variations in the priors, measured in this case by *precision*.

Three classification models are compared, referred to as A, B, and C respectively, where the first uses a principal-component analysis representation (3 components), followed by a mixture of Gaussians classifier (3 mixtures per class), and the second two use the dissimilarity approach [2], using 15 and 50 randomly selected prototypes respectively, and a minimum-distance classifier. A 20-fold randomised hold-out method is used, in which 80% of the data is used in training, and the remainder for testing (cross-validation is not recommended for this dataset (image data), but we use it only for illustration of the principles).In comparing the models, we consider 3 measures:

- $AUPREC$ for $\pi_p = 0.5, 0.1, 0.01$, indicating the integrated *precision* for various skew values.
- $IAUPREC([0.05, 0.20])$, indicating the integrated *precision* for a range of priors $0.05 \leq \pi_p \leq 0.20$. This score is normalised by the area over the range.
- AUC, for reference purposes.

Results (with standard deviation) for the various measures are shown in Table 2.3. Initially, a general observation can be made that the absolute measures indicate that the performance of C is superior to both A and B, and that B is superior to A. We note, however, that there is a large variance in these results, especially of B and C, which makes a firm conclusion hard to draw.

Considering the IAUPREC results, a 2-way ANOVA indicates that only algorithm C is statistically better than A and B (with an F-value of 21.04), and that there is no significant difference between A and B. However, the 3-way ANOVA shows a significance between all 3 models (F-value of 483.85), with C being superior to B, and B being superior to A. This result indicates

[5]Obtain from ftp://ftp.ics.uci.edu/pub/machine-learning-databases

Table 2.3: Summary of experimental results.

Model	(A)	(B)	(C)
$AUPREC(0.5)$	0.554(0.014)	0.775(0.108)	0.803(0.084)
$AUPREC(0.1)$	0.554(0.013)	0.629(0.186)	0.781(0.082)
$AUPREC(0.01)$	0.552(0.013)	0.487(0.245)	0.734(0.075)
$IAUPREC$	0.554(0.013)	0.642(0.177)	0.783(0.082)
AUC	0.943(0.005)	0.825(0.046)	0.905(0.019)

that the 3-way ANOVA is more sensitive to model differences since it directly incorporates the variance due to the priors.

Performing a 3-way ANOVA on the $AUPREC$ measures for the 3 different prior values shows that model C is indeed the best, significantly better than both B and A. Similarly, B is significantly better than A over all 3 the priors. Another observation that can be made for the three $AUPREC$ measures is that models A and C remain very stable with respect to a change in the skew, whereas model B is sensitive to skew. This is a very important result, since for a balanced case, models B and C result in similar performance ($AUPREC$ scores of 0.775(0.108) and 0.803(0.084) respectively). For the case in which $\pi_p = 0.01$, the $AUPREC$ performance for B diminishes to 0.487(0.245), whereas C remains relatively stable at 0.734(0.075). The IAUPREC score indicates a lower score over $0.05 \leq \pi_p \leq 0.20$, corroborating the fact that B is sensitive to skew. Model A is extremely insensitive to skew over the range, but because of the high bias, it would probably not be considered. These observations point out the importance of the *precision* analysis proposed here for evaluating imbalanced, imprecise problems. The ANOVA analysis also indicated that there is a significant difference between the $AUPREC(0.5)$ and $AUPREC(0.01)$ measures, but not between the $AUPREC(0.5)$ and $AUPREC(0.1)$, and the $AUPREC(0.1)$ and $AUPREC(0.01)$ measures. These experiments demonstrate practical application of the *precision-recall* analysis, and and also the importance of incorporating the priors as an additional source of variance in hypothesis testing.

2.3.7 Conclusions

In this paper we have presented an extension of the traditional ROC analysis methodology in which we form a 3-dimensional *precision-recall* ROC surface. Here the class priors represent the third dimension as the *precision* measure is dependent on the class priors. This evaluation methodology was demonstrated on a remote sensing application where priors are known to vary over a fixed range. Models were compared using a 3-way ANOVA test in order to incorporate the priors as an additional source of variation. Experiments showed that the incorporation of the priors results in a more sensitive hypothesis test than the 2-way ANOVA test. This demonstrated the efficacy of this approach in highlighting classifiers that are stable over variations in the priors, and so are

suitable for application in imprecise environments.

Acknowledgements: This research is/was supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs, The Netherlands.

Bibliography

[1] A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[2] R.P.W. Duin, D. de Ridder, and D.M.J. Tax. Experiments with object based discriminant functions; a featureless approach to pattern recognition. *Pattern Recognition Letters*, 18(11-13):1159–1166, 1997.

[3] P. Flach. The geometry of roc space: understanding machine learning metrics through roc isometrics. *ICML-2003, Washington DC*, pages 194–201, 2003.

[4] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2nd edition, 1990.

[5] W. Highleyman. Linear decision functions, with application to pattern recognition. *Proc. IRE*, 49:31–48, 1961.

[6] M. Kubat, R. Holte, and S. Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning, Special issue on applications of machine learning and the knowledge discovery process*, 30:195–215, 1998.

[7] T.C.W. Landgrebe, P. Paclík, D.M.J. Tax, S Verzakov, and R.P.W Duin. Cost- based classifier evaluation for imbalanced problems. *SSSPR+SPR, Lisbon, Portugal*, pages 762–770, 2004.

[8] P. Latinne, M. Saerens, and C. Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities may significantly improve classification accuracy: Evidence from a multi-class problem in remote sensing. *ICML-2001*, pages 298–305, 2001.

[9] C. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, 3(4), 1978.

[10] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42:203–231, 2001.

[11] C. J. Van Rijsbergen. *Information Retrieval*. London, Butterworths, second edition, 1979.

[12] R.E. Walpole. *Introduction to Statistics*. Macmillan, New York, third edition, 1982.

[13] G.M. Weiss. The effect of small disjuncts and class distribution on decision tree learning. *PhD. Dissertation, Department of Computer Science, Rutgers University*, May 2003.

Chapter 3

Extending ROC analysis to multiclass problems

3.1 Overview

Since 2-class classification problems are only a subset of the plethora of problems that may be encountered in pattern recognition, generalising the ROC confers the respective benefits to a much wider range of problems. Active research in this area has only begun quite recently, lacking both a good formalism of the multiclass generalisation (this chapter), and being severely restricted by computational constraints (next chapter).

In this chapter, multiclass ROC surfaces are generated, and applied to two popular ROC analyses. The first ROC generalisation considers the extension of ROC-based Neyman-Pearson optimisation [1]. In the 2-class case, a trained classifier has 2 possible errors, namely the false-negative and false-positive rates. With Neyman-Pearson optimisation, the ROC is used to fix one of the errors, and the model achieving the minimum error on the corresponding error is chosen. In the C-class case, there are $C^2 - C$ different errors (and C performances). A simple algorithm is proposed for fixing any of the classifier outputs, proceeding to minimise the remaining outputs based on the feasible region on the ROC hypersurface. A solution is only guaranteed if one output is specified, but the approach may nevertheless be practically useful in some cases where the system is able to achieve multiple specifications. The second ROC tool generalised to the multiclass case is an extension of the well known Area Under the ROC (AUC) measure. The approach involves integrating the ROC hypersurface, resulting in a volume that indicates a classifier's overall performance (over all operating points/conditions). Similar to the 2-class case, poor classifiers result in lower scores, whereas good classifiers result in high scores. In this contribution, a simplified approach is proposed which can be applied to any classifier. The approach considers only inter-class performance, resulting in C-dimensional volumes. Importantly, the bounds between random and perfect classification as a function of C are computed.

Future work in this area includes the full extension of the volume under the ROC, but a warning has been pointed out in [2]. The authors show that the full extension may not be a useful measure because, as C increases, the measure becomes less sensitive to the difference between a poor and good classifier. The simplified approach proposed in this chapter does not suffer from this limitation. Nevertheless, work in [4] has taken a number of steps in performing this task.

Other research in this area which could be important for the future is the efficient generation of the classifier operating weights, used to compute the multiclass ROC. The approach in the 2-class case can be performed optimally for a given dataset, using actual samples to generate as few as possible (and as many as necessary) weights (see [3]). This approach is valid because classifier outputs are monotonically increasing. However, this is not the case for multiclass problems. The approach taken in this paper has been to generate a grid of weights (logarithmically), which ensure very low to very high weightings. However, different classifiers result in different output scalings. The manner in which the output is scaled could impact the effectiveness/efficiency of a partic-

ular weighting scheme. Research on this topic should consider various classifier types, and possible re-scaling (non-linearly) of outputs. This research could lead to much more efficient multiclass ROC calculations.

Another popular analysis used in the 2-class case via ROC analysis is cost-sensitive optimisation. The extension is however natural, and trivial, given an ROC hypersurface, essentially involving consideration of the overall risk/loss at each operating point.

Bibliography

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

[2] D.C. Edwards, C.E. Metz, and R.M. Nishikawa. The hypervolume under the ROC hypersurface of "near-guessing" and "near-perfect" observers in N-class classification tasks. *IEEE Transactions on Medical Imaging*, 24(3):293–299, March 2005.

[3] T Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters, Special issue on ROC analysis*, 27:861–874, 2005.

[4] C. Ferri, J. Hernandez-Orallo, and M.A. Salido. Volume under the roc surface for multi-class problems. *Proc. of 14th European Conference on Machine Learning*, pages 108–120, 2003.

3.2 On Neyman-Pearson optimisation for multiclass classifiers

This section has been published as 'On Neyman-Pearson optimisation for multiclass classifiers', by T.C.W. Landgrebe and R.P.W. Duin, in *Sixteenth Annual Symposium of the Pattern Recognition Assoc. of South Africa*, November 2005

Abstract

Typically two procedures are used in optimising classifiers. The first is cost-sensitive optimisation, in which given priors and costs, the optimal classifier weights/thresholds are specified corresponding to minimum loss, followed by model comparison. This procedure extends naturally to the multiclass case. The second is Neyman-Pearson optimisation, in which costs may not be certain, and the problem involves specification of one of the class errors, which subsequently fixes the corresponding error (in the two class case), followed by comparisons between different models. This optimisation is well understood in the two-class case, but less so in the multiclass case. In this paper we study the extension of Neyman-Pearson optimisation to the multiclass case, involving specifying various classification errors, and minimising the others. It is shown empirically that the optimisation can indeed be useful for the multiclass case, but obtaining a viable solution is only guaranteed if a single error is specified. Specifying more than one error may result in a solution depending on the data and classifier, which is determined via a multiclass ROC analysis framework.

3.2.1 Introduction

In statistical pattern recognition, a typical design procedure involves gathering representative data for each class, and estimating model parameters to derive a discrimination function (e.g. density estimation, support vector classification), as well as a suitable representation (e.g. feature extraction, feature selection [1]). Once a suitable model is found, the next step is to optimise the various classification weights/thresholds. This optimisation is defined by the nature of the problem at hand. In some situations, the optimisation can be posed as a loss-minimisation problem. In this case classification costs are known, and the respective loss can be computed for different classification weights by summing confusion matrix errors, weighted by the respective costs and priors. This is commonly known as cost-sensitive optimisation [1], [7].

In other situations, referring specifically to the 2-class case, precise costs may be unknown, and a different optimisation strategy needs to be taken. Two types of classification errors occur in the 2-class case, namely the false negative rate (FN_r), consisting of class 1 errors misclassified as class 2, and the false positive rate (FP_r) in the opposite case. In this situation, it is often desirable to specify a fixed FN_r or FP_r, and select a model with the corresponding lowest FP_r or FN_r. This is referred to as Neyman-Pearson optimisation[1]. The optimisation is in selecting the best model. A well-known classifier analysis approach that is useful in this context is receiver operator characteristic (ROC) analysis [6], consisting of a graph representing all possible classification conditions as the classification weights are varied. It is important to note that in the 2-class case, any FN_r or FP_r specification can be achieved, and a corresponding weight obtained. This Neyman-Pearson design is useful in a number

[1] A more fundamental formalisation and derivation of Neyman-Pearson theory in a detection context can be found in [4], with application in a classification sense in [1].

of areas such as detection problems, and medical decision making.

In the multiclass case, several possible classification outputs result, with $C^2 - C$ interclass errors, and C intraclass correct classifications, in a C-class problem. In this situation, it has been shown that both cost-sensitive optimisation and Neyman-Pearson optimisation extend theoretically [2], involving the use of multiclass ROC hypersurfaces. In the case of Neyman-Pearson optimisation, it has thus been shown that specifying a particular classification error in a C-class problem is achievable, with the subsequent objective to minimise all other $C^2 - C$ classification errors. However, a practical situation may demand the specification of a number of classification errors, and subsequent minimisation of remaining classification errors. This type of optimisation could be applicable to areas such as medical decision making involving multiple diseases, or remote sensing, in which the objective is to identify various types of terrain, and minimise the false positive rates with respect to the desired classes of all other terrain types. This type of analysis has not yet (to our knowledge) been studied. In this paper we formalise multiclass ROC analysis, allowing for an implementation of a multiclass Neyman-Pearson optimisation procedure. Extendibility and limitations are identified, primarily discussing the fact that a feasible point on the ROC hypersurface is only guaranteed if just one interclass classification error is specified. However, some experiments show that in practical situations, specifying a number of classification outputs does result in a feasible solution. This is a very interesting result, which may have a large potential for Neyman-Pearson type problems in the multiclass case.

The paper is structured as follows: Section 3.2.2 formalises multiclass classification, allowing for derivation of the various inter- and intra-class outputs inherent to multiclass classifiers. The foundation of the Neyman-Pearson optimisation is a multiclass ROC framework, defined in Section 3.2.3. Neyman-Pearson optimisation using ROC analysis is then discussed in Section 3.2.4, with some experiments to demonstrate the optimisation in realistic situations in Section 3.2.5. A final discussion and conclusions are given in Section 3.2.6.

3.2.2 Formalisation of multiclass classification

Consider a multiclass problem with C classes, $\omega_1, \omega_2, \ldots, \omega_C$, with input data $\mathbf{x}$, and d dimensions. The objective of a multiclass classifier $f(\mathbf{x})$ is to discriminate between the various classes as well as possible, according to the requirements of the problem. The classifier is usually trained based on independent training data. Many strategies are possible, but the outcome is typically a vector of continuous values, with higher values supporting higher confidence (e.g. probability, distance to a decision boundary, support vector etc.) in particular classes. For class i, the classifier output is written as $f(\omega_i|\mathbf{x})$. In the density-based case, this would be the posterior estimate for the respective class. Irrespective of the classification type, the class assignment is typically:

$$argmax_{i=1}^{C} f(\omega_i|\mathbf{x}) \tag{3.1}$$

For example in Figure 3.1, a scatterplot is shown of a 5-class synthetic example, together with the multiclass decision boundary of a Bayes quadratic classifier.

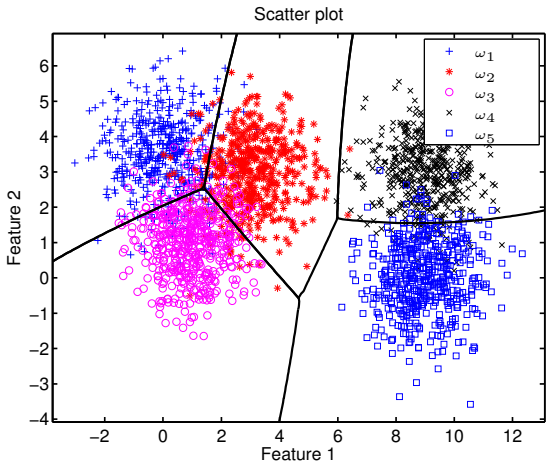


Figure 3.1: Scatter plots of a synthetic 5-class problem, illustrating the decision boundaries (degrees of freedom) for one operating point.

An observation that can be made is that there are a number of Degrees Of Freedom (DOF) that can be used to adjust the classifier (analogous to *thresholds*). In fact, there are $C-1$ degrees of freedom in a $C-$class problem. Thus in the 2-class case, there is only one DOF, and in the 10-class case, there are 9 DOF to optimise the classifier.

When evaluating a classifier, both intraclass outputs (correct classifications), and interclass classifications (between-class errors) are of interest. These are specified by a confusion matrix CM, with a size C^2. Thus the number of errors increases quadratically with increasing C. The CM is typically constructed by applying an independent test set to a trained classifier. In the 2-class case, only 2 interclass errors occur, namely the familiar False Negative rate (FN_r) and False Positive rate (FP_r), with respect to one of the classes. ROC analysis involves inspecting the interplay between these two errors as a function of the single weight/threshold. The CM is defined in Table 3.1. The output between class i and j is denoted $cm_{i,j}$. CM outputs are usually normalised by the absolute number of objects N_i per class ω_i, $N = [N_1, N_2, \ldots, N_C]^T$, resulting in the normalised confusion matrix Ξ, where each element is now referenced as $\xi_{i,j}$, and $\xi_{i,j} = \frac{cm_{i,j}}{N(i)}$. We now consider the computation of each element in Ξ via an example. Consider the class-conditional distributions in Figure 3.2, consisting of five Gaussian-distributed classes $\omega_1, \omega_2, \ldots, \omega_5$, with means occurring at $\mu_1 = -6, \mu_2 = -3, \mu_3 = 0, \mu_4 = 6, \mu_5 = 9$, and unit variance. The respective priors are assumed to be equal. The class-conditional density of class i is denoted $p(\mathbf{x}|\omega_i)$, and the prior $p(\omega_i)$.

		estimated				
		ω_1	ω_2	$\ldots$	ω_C	
true	ω_1	$cm_{1,1}$	$cm_{1,2}$	$\ldots$	$cm_{1,C}$	N_1
	ω_2	$cm_{2,1}$	$cm_{2,2}$	$\ldots$	$cm_{2,C}$	N_2
	$\ldots$					
	ω_C	$cm_{C,1}$	$cm_{C,2}$	$\ldots$	$cm_{C,C}$	N_C

Table 3.1: A multi-class confusion matrix.

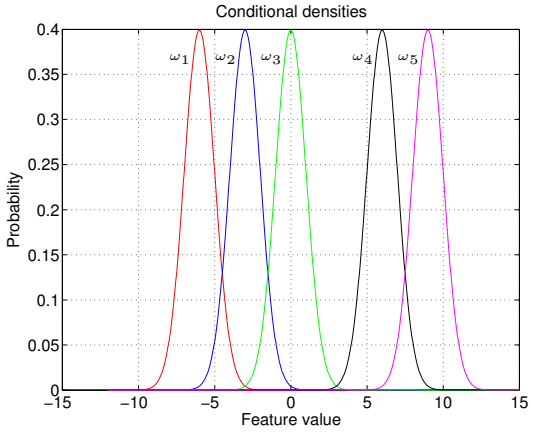


Figure 3.2: Probability density functions for the 5-class example with known distributions.

This example results in a 5×5 element confusion matrix. In order to compute each confusion $\xi_{i,j}$ (the percentage of of ω_i misclassified as ω_j), the following integration is performed:

$$\xi_{i,j}(\mathbf{x}) = p(\omega_i) \int p(\mathbf{x}|\omega_i) I_{ij}(\mathbf{x}) dx \tag{3.2}$$

The indicator function $I_{ij}(\mathbf{x})$ specifies the relevant domain:

$$I_{ij}(\mathbf{x}) = \begin{cases} 1 \text{ if } p(\omega_j|\mathbf{x}) > p(\omega_k|\mathbf{x}) \ \forall k, \\ k \neq j,\ i \neq j \\ 1 \text{ if } p(\omega_i|\mathbf{x}) > p(\omega_k|\mathbf{x}) \ \forall k, \\ k \neq i,\ i = j \\ 0 \text{ otherwise} \end{cases} \tag{3.3}$$

Equation 3.2 allows any confusion matrix output to be computed, generalised for both diagonal elements (performances), and off-diagonal elements (errors).

3.2.3 Multiclass ROC analysis

Referring again to Figure 3.1, the plot shows only a single operating point, corresponding to a single weight setting. In fact, any combination of weightings results in a different operating point (the challenge in multiclass optimisation is in understanding the relation between a weight modification and the corresponding alteration of the confusion matrix). Application of this weighting Φ involves modification of Equation 3.1, in which the class assignment is now based on each output $f(\omega_i|\mathbf{x})$, multiplied by a corresponding weight, denoted ϕ_i, resulting in:

$$argmax_{i=1}^{C} \phi_i f(\omega_i|\mathbf{x}) \tag{3.4}$$

The concept of classifier optimisation can be formalised as the process by which the optimal set of weights is found to suit the problem at hand. ROC analysis involves generation of a hypersurface consisting of all possible combinations of Φ, where $\Phi = [\phi_1, \phi_2, \ldots, \phi_{C-1}, 1 - \phi_1]$. A multiclass ROC consists of $C^2 - C$ dimensions (diagonal elements are superfluous), which can be constructed using a similar equation to 3.2. In this case, each output between class i and j is weighted by ϕ_i as follows:

$$\xi_{i,j}(\mathbf{x}|\Phi) = \phi_i p(\omega_i) \int p(\mathbf{x}|\omega_i) I_{ij}(\mathbf{x}|\Phi) dx \tag{3.5}$$

The indicator function $I_{ij}(\mathbf{x}|\Phi)$ is as in 3.3, except each posterior is multiplied by the corresponding class weight. Note that there are only $C-1$ weights, and thus $\phi_C = 1 - \phi_1$.

Consider the 2-class case between ω_1, and ω_2, in which a weighting ϕ is applied to obtain the most appropriate threshold. In this case, the classifier output can be written as:

$$p(\mathbf{x}|\phi) = [\phi p(\omega_1|\mathbf{x}) \; (1 - \phi) p(\omega_2|\mathbf{x})] \tag{3.6}$$

For $0 \leq \phi \leq 1$, $\xi_{1,2}$ and $\xi_{2,1}$ vary across all possible combinations, resulting in the ROC plot. In the multiclass case ($C > 2$), Equation 3.5 can be used to construct a multiclass ROC, resulting in a $C^2 - C$ dimensional surface. For example, in the 5-class Gaussian example, a 4-D grid of weights was computed ($C - 1$ DOF), and a step resolution of 30 was chosen, resulting in $8.1e5$ weight combinations. Application of Equation 3.5 resulted in a 20 dimensional surface. Even though this surface cannot be visualised, for demonstration purposes, Figure 3.3 shows the ROC between the dimensions $\xi_{1,2}$, $\xi_{2,1}$, and $\xi_{2,3}$.

3.2.4 Neyman-Pearson optimisation

Classifier optimisation in a Bayesian framework involves estimates (or given) class conditional densities (pdf's), denoted $f(\mathbf{x}|\omega_i), \forall i$, prior estimates $\pi_i, \forall i$, and misclassification costs corresponding to each off-diagonal output of the confusion matrix CM, denoted $c_{ij}, i \neq j$. The optimisation involves deriving

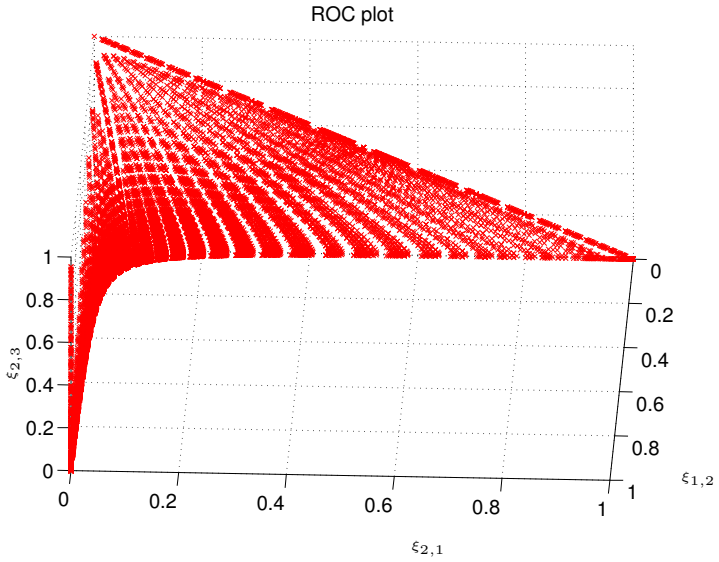


Figure 3.3: Plotting $\xi_{1,2}$, $\xi_{2,1}$, and $\xi_{2,3}$ for the example in Figure 3.2 as a function of Φ.

the optimal weight vector Φ such that the overall system loss is minimised, where the loss is computed via:

$$L = \sum_{i=1}^{C} \frac{\phi_i \pi_i}{N_i} \left(\sum_{j=1, i \neq j}^{C} cm_{i,j} c_{ij} \right) \tag{3.7}$$

In some problems (such as detection problems and medical decision making), respective costs cannot be defined, and the cost-sensitive optimisation procedure cannot be taken. However, it is assumed that the individual classes can be modelled in some way, e.g. pdf estimates $f(\mathbf{x}|\omega_i), \forall i$. In the 2-class case, this implies that an ROC curve can be estimated between $\xi_{1,2}$ and $\xi_{2,1}$ (referring to the previous example), the false negative, and false positive rates respectively, written as a function of the weight ϕ as (with population priors π_i $\forall i$ estimated):

$$\begin{aligned} \xi_{1,2}(\mathbf{x}|\phi) &= \phi \pi_1 \textstyle\int f(\mathbf{x}|\omega_1) I_{12}(\mathbf{x}|\phi) dx \\ \xi_{2,1}(\mathbf{x}|\phi) &= \phi \pi_2 \textstyle\int f(\mathbf{x}|\omega_2) I_{21}(\mathbf{x}|\phi) dx \end{aligned} \tag{3.8}$$

In Neyman-Pearson optimisation, either $\xi_{1,2}$ or $\xi_{2,1}$ is fixed at a specified value α. The optimisation then involves computing a value for the weighting ϕ such that the specification holds, and the dependent variable is obtained. In the 2-class case, optimisation occurs only across models. This is illustrated on the ROC plot in Figure 3.4 (plotting the false negative rate against false positive rate). In this example, $\xi_{1,2}$ is fixed at 10.00%. The ROC curve is a useful tool in this case, immediately resulting in the corresponding $\xi_{2,1}$, which is approximately 26%. The ϕ resulting in this point can then be used as the

optimal Neyman-Pearson threshold (Note that each point on the ROC plot corresponds to some ϕ value, and in the multiclass ROC, each point on the ROC hypersurface corresponds to a C dimensional weight vector Φ).

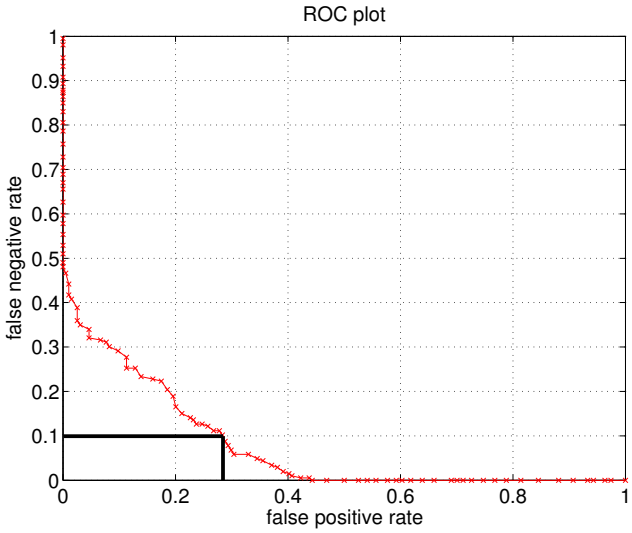


Figure 3.4: ROC plot example illustrating Neyman-Pearson design. In this case the false negative rate has been specified at 10.00%.

The Neyman-Pearson optimisation is well studied and extensively used in 2-class problems, but this is not the case in the multiclass context. A practical implementation involves the use of multiclass ROC analysis which is also a relatively new research area (see some recent works in [3],[5], [2]). This was also formalised in Section 3.2.3. Recently, the theoretical extension of Neyman-Pearson optimisation to multiclass optimisation was proven in [2], with applicability to multiple-diagnosis in medical decision making. This showed that by fixing a single classification error in the confusion matrix, a solution is guaranteed. Thus the first step involves specifying some classification error, and subsequently to return a Φ corresponding to a minimisation of all other $C^2 - C - 1$ classification errors in Ξ. Section 3.2.5 demonstrates this optimisation in a few experiments.

In this paper, we also wish to generalise the optimisation procedure such that multiple errors in Ξ can be specified, followed by a subsequent minimisation of remaining errors. It is obvious that when specifying more than one error, a solution is not guaranteed (multiple dependencies). However, we argue that in many practical situations, it may still be feasible to specify a number of outputs, and obtain a solution. The usefulness is, however, data and problem dependent. Next, an algorithm is developed that is generalised in the sense that the original Neyman-Pearson optimisation holds (specifying one error), and can also be used to specify multiple errors.

Algorithm 1: Multiclass Neyman-Pearson optimisation
Inputs: ROC resolution $step$, C classes, trained classifier D, error specification matrix M_e, specification index matrix M_I, independent test set $\mathbf{x}_{ts}$
Outputs: Optimal weights Φ_{opt}
1) Construct weight matrix Φ, with resolution $step$
2) Compute $C \times C$ multiclass ROC E, using Equation 3.5, with $step$ resolution, applying $\mathbf{x}_{ts}$ to D, for all Φ
3) $m = 0$
For each row of E i, and column j $(i \neq j)$:
If $(M_I(i,j) = 1$ and $M_e(i,j) > 0)$
$ind_{(m)} = E(i,j,k) \leq M_e(i,j)\ \forall k$)
Increment m
End
4) Find hypersurface regions that fulfil M_e specifications:
$ind_{com} = ind_{(1)} \cap ind_{(2)} \cap \ldots ind_{(m)}$
If size of $ind_{com} = 0$, no solution - specifications not met
5) Minimise all other non-specified errors
$m = 0$
For each row i, and column j of E $(i \neq j)$:
If $(M_I(i,j) = 0)$
$err_{(m)} = \sum(\sum_{i=1}^{C} \sum_{j=1}^{C}(E(i,j,k)\ \forall k)\ i \neq j)$
Increment m
End
Index corresponding to minimum error:
$ind_{min} = index(min(err_{(m)}))$, $\forall m$
Final classifier weights: $\Phi_{opt} = \Phi(ind_{min})$
Return: Φ_{opt}, or no solution

The proposed optimisation algorithm is implemented by the introduction of two $C \times C$ matrices, namely M_I and M_e (many other implementations are possible). The elements of these correspond to confusion matrix outputs, allowing for a direct input of required specifications. The M_I matrix is a binary indicator matrix specifying which errors are to be specified. A 1 at position $M_I(i,j)$ indicates that $\xi_{i,j}$ is to be specified, and a 0 at position $M_I(k,l)$ indicates that $\xi_{k,l}$ is to be minimised. Working in conjunction with M_I is M_e, a matrix used to specify the respective errors as specified in M_I. For example, requiring an error rate lower or equal to 5.00% for $\xi_{3,1}$ would require a $M_I(3,1) = 1$ and $M_e(3,1) = 0.05$.

Algorithm 1 presents a practical procedure that can be used to perform a multiclass Neyman-Pearson optimisation[2].

In step 2, the multi-class ROC is computed, denoted E using a matrix of weights with $C - 1$ columns, as per step 1 (we assume dense sampling). It is convenient to store this matrix in a similar form to the confusion matrix, resulting in a $C \times C \times step^{C-1}$ dimensional matrix. The diagonal elements are ignored, and need not be computed. In step 3, all ROC dimensions corresponding to error specifications are inspected, and the portion of the surface (in that dimension m) fulfilling the specification is stored. The range corresponding to dimension m is stored in $ind_{(m)}$. Note that the *leq* operator is

[2]This algorithm can easily be adapted for the case in which the distributions are known. We focus here on the practical situation in which the true distributions are unknown, and data samples are assumed to originate from the true distribution.

used here since the ROC is discretised. The next step 4 involves intersecting each of these ranges, and identifying common indices (denoted ind_{com}). If no intersection occurs, this implies that the specifications cannot be met (no point on the ROC hypersurface fulfils specifications). If a solution does exist, step 5 involves minimisation of all non-specified errors. This is achieved by summing all non-specified errors for each Φ weighting corresponding to indices ind_{com}. The weighting resulting in the lowest error sum is then chosen as Φ_{opt}.

3.2.5 Experiments

To demonstrate the optimisation in a practical situation, the *Satellite* dataset is considered[3]. This dataset consists of 6435 multi-spectral values of a satellite image, with 36 dimensions (4 spectral bands in a 9 pixel neighbourhood). Six classes have been identified to characterise the topography. These consist of *red soil*, *cotton crop*, *grey soil*, *damp grey soil*, *soil with vegetable stubble*, and *very damp grey soil* classes. In experiments, all the *grey soil* classes are grouped together, resulting in a 4 class problem. As per dataset recommendations, the first 4435 spectra are used as a training set, and the remaining data as the test set.

The first experiment involves training a base classifier on the data. The first 17 principal components are used to represent the spectra, and a Bayes linear discriminant classifier is then trained on this representation. The following normalised confusion results following application of the independent test set:

```
       |   red     cotton     veget     grey
------ |---------------------------------------
red    | 0.9491    0.0000    0.0081    0.0428
cotton| 0.0047    0.8826    0.0986    0.0141
veget  | 0.0553    0.0046    0.7051    0.2350
grey   | 0.0009    0.0000    0.0037    0.9954
```

Following classical Neyman Pearson, we now experiment by only specifying single classification errors, and minimising all others. Two experiments are demonstrated. In the first experiment, $\epsilon_{veget,grey} = 8.00\%$, and in the second, $\epsilon_{cotton,veget} = 5.00\%$, resulting in the following two normalised confusion matrices:

```
       |   red     cotton     veget     grey
------ |---------------------------------------
red    | 0.9572    0.0000    0.0204    0.0224
cotton| 0.0047    0.8826    0.1127    0.0000
veget  | 0.0461    0.0046    0.8756    0.0737
grey   | 0.0009    0.0000    0.0454    0.9537

       |   red     cotton     veget     grey
------ |---------------------------------------
red    | 0.9552    0.0000    0.0000    0.0448
cotton| 0.0047    0.9061    0.0469    0.0423
veget  | 0.0599    0.0046    0.4516    0.4839
grey   | 0.0009    0.0000    0.0009    0.9981
```

[3]UCI repository of machine learning databases, ftp://ftp.ics.uci.edu/pub/machine-learning-databasesx.

As expected, these result in a solution. Comparing with the original normalised confusion matrix, it can clearly be seen that this new operating point has resulted in different compromises between the various classes (which may or may not be acceptable). In the next experiments, we attempt to specify a number of interclass errors, and minimise the remaining ones. Three experiments are undertaken with the following specifications:

1. Specify $\epsilon_{veget,red} = 5.00\%$, $\epsilon_{red,grey} = 5.00\%$, and $\epsilon_{veget,grey} = 5.00\%$.
2. Specify $\epsilon_{veget,red} = 5.00\%$, $\epsilon_{cotton,veget} = 15.00\%$, and $\epsilon_{veget,grey} = 5.00\%$.
3. Specify $\epsilon_{veget,red} = 5.00\%$, $\epsilon_{cotton,veget} = 10.00\%$, and $\epsilon_{veget,grey} = 5.00\%$.

The optimisation is successful in the first two experiments, but fails in the third. This implies that the specifications cannot be achieved by the classifier in this case. The first two cases result in the following respective normalised confusion matrices:

```
       |   red     cotton     veget     grey
-------|---------------------------------------
red    | 0.9593    0.0000    0.0224    0.0183
cotton| 0.0047    0.8826    0.1127    0.0000
veget  | 0.0461    0.0046    0.9171    0.0323
grey   | 0.0009    0.0000    0.1186    0.8804

       |   red     cotton     veget     grey
-------|---------------------------------------
red    | 0.9593    0.0000    0.0224    0.0183
cotton| 0.0047    0.8592    0.1362    0.0000
veget  | 0.0461    0.0046    0.9171    0.0323
grey   | 0.0009    0.0000    0.1186    0.8804
```

These normalised confusion matrices should be compared to the original one. It can be seen in the first case that the output $\xi_{veget,veget}$ has improved from around 70% to over 90%, but $\xi_{grey,grey}$ is around 11% lower. Note that many solutions are often possible. These specifications were also not met in the first examples, in which single errors were specified. These experiments (which are limited due to space constraints) demonstrate the potential and practicality of the proposed approach. Once a multiclass ROC has been computed, a practitioner can quickly and easily enter and modify a specification via the matrices M_I and M_e.

3.2.6 Conclusion

This paper considered the applicability of Neyman-Pearson optimisation to multiclass problems. This is a well-studied and extensively used technique in 2-class problems (stemming from detection applications), applicable in situations where costs cannot be defined, and it is more practical to specify a fixed true- or false- positive rate. ROC analysis is a tool facilitating this optimisation, allowing for a direct query of the corresponding classification threshold/weight. In the multiclass case, several possible classification outputs result. Work performed in [2] showed that Neyman-Pearson optimisation does hold in

the multiclass case, in which case a single output/error is specified. A more practical multiclass scenario may involve the necessity to specify multiple error rates. This paper investigated this empirically, with results showing that this is possible, but a solution is not guaranteed. Since the optimisation hinges on ROC analysis, a multiclass ROC framework was developed, and an algorithm designed to perform the optimisation. The algorithm attempts to find regions on the ROC hypersurface that meets the specifications, and then subsequently minimises all other classification errors. The algorithm terminates in the case that a solution cannot be found. Some real experiments demonstrated the potential of this approach. On-going research is focused on obtaining efficient representations of the ROC hyper-surface for large C problems, which remains a significant challenge to multiclass design.

Acknowledgements: This research is/was supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs, The Netherlands.

Bibliography

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

[2] D.C. Edwards, C.E. Metz, and M.A. Kupinski. Ideal observers and optimal ROC hypersurfaces in N-class classification. *IEEE Transactions on Medical Imaging*, 23(7):891–895, July 2004.

[3] C. Ferri, J. Hernandez-Orallo, and M.A. Salido. Volume under the roc surface for multi-class problems. *Proc. of 14th European Conference on Machine Learning*, pages 108–120, 2003.

[4] D. Kazakos and P. Papantoni-Kazakos. *Detection and Estimation, ISBN 0-7167-8181-6*. Computer Science Press, 1st edition, 1990.

[5] N. Lachiche and P. Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. *Proc. 20th Int. Conf. on Machine Learning, Washington DC*, pages 416–423, 2003.

[6] C. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, 3(4), 1978.

[7] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42:203–231, 2001.

3.3 A simplified volume under the ROC hypersurface

This section has been accepted as 'A simplified volume under the ROC hypersurface', by T.C.W. Landgrebe, and R.P.W. Duin, in *Transactions of the South African Institute of Electrical Engineers*, 2007, based on the paper published as 'A simplified extension of the Area under the ROC to the multiclass domain', by T.C.W. Landgrebe, and R.P.W. Duin, in *Seventeenth Annual Symposium of the Pattern Recognition Association of South Africa*, November 2006.

Abstract

The Receiver Operator Characteristic (ROC) plot allows a classifier to be evaluated and optimised over all possible operating points. The Area Under the ROC (AUC) has become a standard performance evaluation criterion in two-class pattern recognition problems, used to compare different classification algorithms independently of operating points, priors, and costs. Extending the AUC to the multiclass case is considered in this paper, called the volume under the ROC hypersurface (VUS). A simplified VUS measure is derived that ignores specific intra-class dimensions, and regards inter-class performances only. It is shown that the VUS measure generalises from the 2-class case, but the bounds between random and perfect classification differ, with the lower bound tending towards zero as the dimensionality increases. A number of experiments with known distributions are used to verify the bounds, and to investigate a numerical integration approach to estimating the VUS. Experiments on real data compare several competing classifiers in terms of both error-rate and VUS. It was found that some classifiers compete in terms of error-rate, but have significantly different VUS scores, illustrating the importance of the VUS approach.

3.3.1 Introduction

A very active area in pattern recognition has been the consideration of classifier design and evaluation in less well-defined environments e.g. undefined or varying prior probabilities [16], or poorly defined costs [1]. A primary analysis tool developed for this domain is Receiver Operator Characteristic (ROC) analysis [11], allowing a classifier to be inspected over a range of possible conditions. A popular scalar performance measure that has emerged is the Area Under the ROC (AUC) [3], allowing classifiers to be evaluated independent of priors, costs, and operating points. The AUC measure is however only applicable to the 2-class case. Considering the multiclass extension of this measure has become a topic of interest more recently, often referred to as the Volume Under the ROC hyper-Surface (VUS). Formalisation and computational aspects are more complex, but nevertheless a number of steps have been taken to generalise the AUC. In [15], a simplified VUS is estimated from a multiclass classifier by considering the AUC between each class, and all other classes (a one vs all approach), resulting in a computationally tractable algorithm $O(C)$, where there are C classes. This measure is however inherently dependent on class priors and costs, and ignores higher-order interactions. In [9], a similar estimation of the VUS is proposed that averages the AUC between all pairs of classes, which has a higher complexity of $O((C-1)(C-3)(C-5)\dots 1)$. The exact theoretical extension to the VUS in the 3-class case has been considered in [12] and [4]. In [8] the generalised VUS has been studied, providing calculations/estimations of the performance bounds of the VUS as a function of an increasing number of classes C. This involved comparing performance between perfect (separable) classifiers and random classifiers (random performance). This non-trivial study provides an important step in understanding the VUS performance measure.

A related paper was presented in [6], which 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.

Previous works have not gone into detail as to how the VUS can practically be applied to an arbitrary set of classifiers in realistic scenarios. In this paper we consider the practical implementation of the VUS, applied to the simplified scenario in which the overall class performances are considered, ignoring specific intra- and inter-class errors. This type of simplification restricts the VUS analysis, but nevertheless may be suitable for some problems e.g. where we are still interested in all operating points in terms of overall class performance, but the class to which an erroneous object is assigned is arbitrary (hand-written digit recognition/ face recognition are two possible applications). This simplification ensures that good classifiers tend to result in higher VUS scores than poorer ones, irrespective of C (as will be shown), resulting in an alternative measure in line with the argument in [6]. The approach presented here provides a practical methodology for computing the VUS for problems with low C[4], demonstrated via a number of experiments. In Section 3.3.2 the notation is presented, followed by a brief formalisation of multiclass ROC analysis, and the well-known AUC in Section 3.3.3. In Section 3.3.4 the simplified VUS is presented. First performance bounds are derived as a function of C. A numerical integration procedure is then proposed in order to resample the irregularly-spaced multiclass ROC, allowing for accurate estimations of the VUS. A number of problems involving known distributions are used to verify the bounds and the methodology. In Section 3.3.5 a number of experiments involving real data are presented, demonstrating practical usage of the VUS measure in 3- and 4-class problems. Finally conclusions are presented in Section 3.3.6.

3.3.2 Notation

We use a framework similar to [10], in which observations $\mathbf{x}$ are to be classified into one of C classes, $\omega_1, \omega_2, \ldots, \omega_C$. Each class ω_i has a class-conditional distribution $p(\mathbf{x}|\omega_i)$, and prior probability $P(\omega_i)$. Class assignment is based on Bayes rule, which assigns membership to the highest posterior output:

$$P(\omega_i|\mathbf{x}) = \frac{P(\omega_i)p(\mathbf{x}|\omega_i)}{P(\omega_1)p(\mathbf{x}|\omega_1) + P(\omega_2)p(\mathbf{x}|\omega_2) + \ldots P(\omega_C)p(\mathbf{x}|\omega_C)} \tag{3.9}$$

Thus $\mathbf{x}$ is assigned according to:

$$argmax_{i=1}^{C} P(\omega_i|\mathbf{x}) \tag{3.10}$$

In the practical case in which class conditional distributions are usually unknown, these are typically estimated from representative examples that are

[4]Extension to the high C case remains computationally infeasible, and thus our approach is restricted to low C problems e.g. $C = 3$ to 6. Simpler approaches such as [9] are the only candidates for high C.

assumed to be randomly drawn from the true distribution, and the same framework can be used. A given classifier is analysed in detail via the $C \times C$ dimensional normalised confusion matrix Ξ, in which diagonal elements represent the overall performance of each class, and off-diagonal elements the errors related to each class. Each element (i, j) of Ξ is denoted $\xi_{i,j}$. Ξ can be written as:

		estimated			
		ω_1	ω_2	$\dots$	ω_C
true	ω_1	$\xi_{1,1}$	$\xi_{1,2}$	$\dots$	$\xi_{1,C}$
	ω_2	$\xi_{2,1}$	$\xi_{2,2}$	$\dots$	$\xi_{2,C}$
	$\vdots$	$\vdots$		$\ddots$	
	ω_C	$\xi_{C,1}$	$\xi_{C,2}$	$\dots$	$\xi_{C,C}$

Table 3.2: Defining the multi-class normalised confusion matrix Ξ.

Each element $\xi_{i,j}$ is computed as follows:

$$\xi_{i,j} = p(\omega_i) \int p(\mathbf{x}|\omega_i) I_{ij}(\mathbf{x}) dx \tag{3.11}$$

The indicator function $I_{ij}(\mathbf{x})$ specifies the relevant domain (with the second line specifying performances on the diagonal elements):

$$I_{ij}(\mathbf{x}) = \begin{cases} 1 \text{ if } p(\omega_j|\mathbf{x}) > p(\omega_k|\mathbf{x}) \ \forall k, k \neq j, \ i \neq j \\ 1 \text{ if } p(\omega_i|\mathbf{x}) > p(\omega_k|\mathbf{x}) \ \forall k, k \neq i, \ i = j \\ 0 \text{ otherwise} \end{cases} \tag{3.12}$$

In the practical case, $\xi_{i,j}$ is estimated via representative test sets, counting the number of objects classified to each element, normalised by the number of objects in that class.

3.3.3 Multi-class ROC analysis

It is important to understand that the confusion matrix actually only indicates the performance of a trained classifier at a single operating point i.e. different operating points result in different confusion matrices. The operating point is varied by weighting the posterior output of the classifier by the vector $\Phi = [\phi_1, \phi_2, \dots, \phi_C], \phi_i > 0, \forall i$, which is analogous to classifier thresholds. Thus Equation 3.10 is modified as $argmax_{i=1}^{C} \phi_i P(\omega_i|\mathbf{x})$. All combinations of Φ result in all possible operating points of the classifier, which is the multiclass ROC. Note that there are in fact only $(C-1)$ degrees of freedom for a trained classifier, so one weight can be held constant, or normalised by the others. After applying all combinations of Φ, a C^2-dimensional operating characteristic results, with each confusion matrix element attributed to a new dimension. Note that only $(C^2 - C)$ dimensions are required, since:

$$\epsilon_{i,i} = 1 - \sum_{j=1, j \neq i}^{j=C} \epsilon_{i,j} \tag{3.13}$$

The two class case is very well known, with two off-diagonal elements resulting ($\xi_{1,2}$ and $\xi_{2,1}$, popularly known as the false negative- and false positive-rates), and two diagonal elements ($\xi_{1,1}$ and $\xi_{2,2}$, the true positive and true negative-rates). This operating characteristic has well understood characteristics and bounds [3], [16]. Varying the classifier threshold results in a $1D$ ROC curve. Figure 3.5 show ROC plots for three different scenarios, ranging from a perfect/separable classifier (A), to a classifier with some overlap (B), and finally to the random classification case (C). Considering the area consumed by each classifier allows performance to be inspected independent of priors, costs, and operating points. In this 2-class case, perfect classification results in a larger area, bounded by 1, and poor classification in a smaller area, bounded by 0.5 (since the random classifier bisects the unit square). This area is known as the Area Under the ROC (AUC). Note that traditionally the ROC is plotted between $\xi_{1,1}$ and $\xi_{2,1}$, but Figure 3.5 results in an equivalent performance measure, and is extensible to the multiclass case.

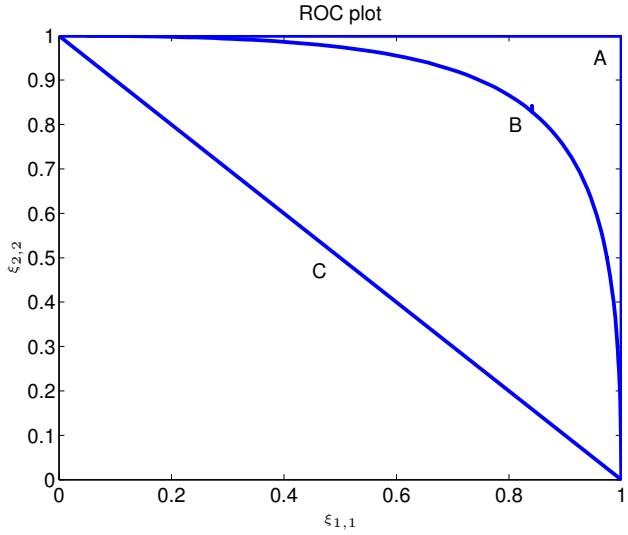


Figure 3.5: Comparing 3 different 2-class ROC plots. 'A' depicts perfect classification, 'B' is a classifier with some overlap, and 'C' is a random classifier.

The AUC can be written as:

$$AUC = \int \xi_{2,2} d\xi_{1,1} \tag{3.14}$$

The AUC can be applied to the realistic scenario by a numerical integration scheme. This work uses the trapezoidal integration rule. The AUC can also be estimated by counting the number of times two arbitrary objects in the test set from both classes are correctly ranked by the classifier, and normalising.

3.3.4 Simplified Volume Under the ROC

Extending the AUC to the multiclass case, i.e. the volume under the ROC hypersurface, can be achieved by measuring the volume bounded by the operating characteristic. In this case we consider only ROC dimensions pertaining to diagonal elements of the confusion matrix. The simplified VUS can be written as:

$$VUS = \int \dots \int \int \xi_{C,C} d\xi_{C-1,C-1} d\xi_{C-2,C-2} \dots d\xi_{1,1} \tag{3.15}$$

Thus the simplified measure considers the $C-$dimensional operating characteristic of a $C-$dimensional problem. This measure allows a classifier to be evaluated over all operating points responsible for the ROC dimensions corresponding to the diagonal confusion matrix elements. If these performances only are considered, the VUS is similar to the AUC in that better classifiers will result in a high VUS, and poorer classifiers in a lower score. However, before the VUS is blindly applied, it is important to characterise and understand the performance bounds between random and perfect classifiers.

Bounds as a function of dimensionality

Considering the 3-class case first, the simplified ROC dimensionality is 3, between the dimensions $\xi_{1,1}, \xi_{2,2}$, and $\xi_{3,3}$. A random classifier produces the ROC depicted in Figure 3.6. A more effective classifier (or more separable problem) is depicted in Figure 3.7, showing how the VUS increases.

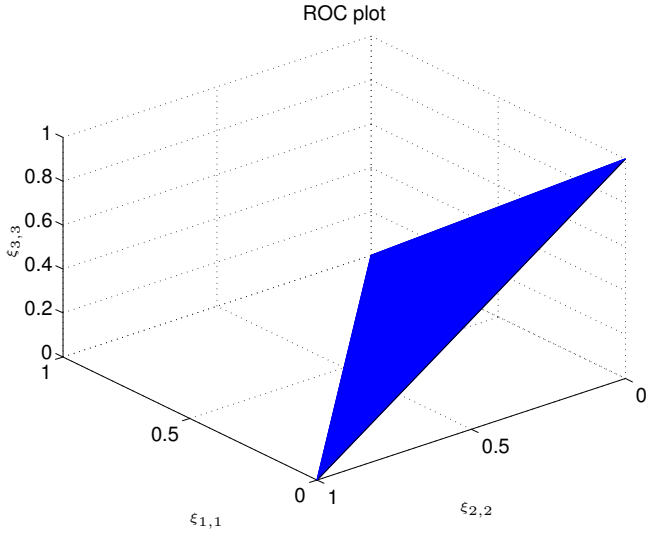


Figure 3.6: Random classification performance of the simplified 3-class ROC.

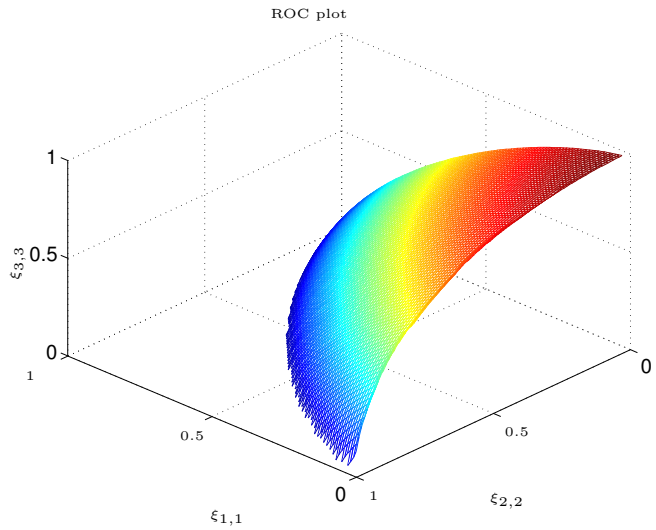


Figure 3.7: ROC plot for a 3-class problem with partially overlapping distributions.

In fact, the VUS approaches 1.0 as the classification becomes perfect. The VUS occupied by the random classifier can be found geometrically by computing the volume of the tri-rectangular tetrahedron formed under the surface, which is simply $\frac{1}{6}\xi_{1,1}\xi_{2,2}\xi_{3,3} = \frac{1}{6}$. Thus the bound has altered from $\frac{1}{2}$ in the two-class case, to $\frac{1}{6} = 0.16666$ in the 3-class case. Generalising the bounds to C classes is more difficult geometrically. A more extensible approach is to formalise the random ROC as a hyper-polyhedron, as proposed in [8]. Each vertex v_i of the hyper-polyhedron can easily be defined as (note that the origin is always included as a vertex, and there are C points per vertex):

$$\begin{array}{lccccc} v_1 & 0 & 0 & 0 & \dots & 0 \\ v_2 & 1 & 0 & 0 & \dots & 0 \\ v_3 & 0 & 1 & 0 & \dots & 0 \\ v_4 & 0 & 0 & 1 & \dots & 0 \\ & \vdots & & & & \\ v_{C+1} & 0 & 0 & 0 & \dots & 1 \end{array} \tag{3.16}$$

As in [8], the optimised QHull [2] algorithm is used to estimate the volume occupied by the hyper-polyhedron. The following lower bounds result, up to $C = 12$, showing how the lower bound approaches zero with an increasing C. In fact, it can be seen that the lower bound is $\frac{1}{C!}$, which is proven in Appendix 3.3.7[5].

[5]The bounds of the simplified VUS suggest this method is a good alternative to the true

C	Estimated VUS
2	0.50000000001826
3	0.16666666668765
4	0.04166666667598
5	0.00833333333563
6	0.00138888888932
7	0.00019841269853
8	0.00002480158732
9	0.00000275573193
10	0.00000027557319
11	0.00000002505211
12	0.00000000208768

(3.17)

Estimating the VUS for general classifiers

In the practical situation in which a sparse set of points are given, representing the multiclass ROC, a different approach is required. Since the ROC surface is derived by the nature of the problem and classifier, it cannot be computed analytically. A more appropriate approach to estimating the VUS is to use a numerical integration approach. The inherent uneven sampling of the ROC is converted to an even form via linear resampling and interpolation. The trapezoidal rule is then used to estimate the volume (in $C-$dimensions), with the following results as a function of r, the number of ROC steps used:

C	r	VUS estimation	Actual VUS
3	50	0.1667014	0.1666666
3	100	0.1666752	0.1666666
4	50	0.0417014	0.0416667
4	100	0.0416752	0.0416667
5	50	0.0083507	0.0083333
5	100	0.0083376	0.0083333
6	20	0.0014275	0.0013889
6	40	0.0013980	0.0013889

These results show that the numerical integration approach provides a good approximation of the true VUS, and that as expected a higher step size results in higher accuracy.

Experiments with known distributions

In order to judge the numerical VUS approach and verify the bounds, a number of controlled experiments are conducted, consisting of generated Gaussian classes with known parameters. The first set of experiments consist of 3-class

unsimplified VUS (regarding the argument given in [6] pertaining to poor resolution between perfect and random classifiers for high dimensions, bringing the validity of the VUS into question). This is because in the simplified case for high C, good classifiers tend to 1, and poor ones tend to 0.

Gaussian problems with classes ω_1, ω_2, and ω_3, in which the means are varied, and the variances held at unity. The means are varied such that the problems range from near-separable problems, to near-random. Similarly the second set of experiments involve varying the means of 4 Gaussian classes. Tables 3.3 and 3.4 depict the results for the 3- and 4-class cases respectively, also showing r (a higher resolution was required as the distributions approached complete overlap). In Figure 3.8, the distributions used in the 2nd and 4th 4-class experiments are shown, demonstrating how class overlap was increased.

Means	**r**	**VUS est.**
$-0.05; 0.0; 0.05$	200	0.16876
$-0.3; 0.0; 0.3$	100	0.24140
$-0.5; 0.0; 0.5$	100	0.31428
$-1.0; 0.0; 1.0$	100	0.51214
$-1.5; 0.0; 1.5$	100	0.70597
$-4.0; 0.0; 4.0$	100	0.98582

(3.18)

Table 3.3: Results for 3-class experiments with known distributions.

Means	**r**	**VUS est.**
$-0.15; -0.05; 0.05; 0.15$	70	0.05688
$-0.75; -0.25; 0.25; 0.75$	50	0.07782
$-1.00; -0.33; 0.33; 1.00$	50	0.19972
$-1.50; -0.50; 0.50; 1.5$	50	0.33097
$-2.25; -0.75; 0.75; 2.25$	50	0.57990
$-3.00; -1.00; 1.00; 3.0$	50	0.75451

(3.19)

Table 3.4: Results for 4-class experiments with known distributions.

These experiments verify that the VUS approach used does make intuitive sense, since it can be seen that as the problems vary from the separable to the random case, the VUS decreases accordingly. For highly overlapping cases, the two sets of experiments demonstrate a VUS that approaches the predicted lower bounds.

3.3.5 Experiments

The VUS methodology is demonstrated in real settings by comparing a number of competing classifiers over a number of different problems. The first group of experiments consist of 3-class problems, with the following datasets used: *Banana* is a 2-dimensional dataset consisting of a Banana-shaped class [5], a Gaussian distributed class, and a bimodal Gaussian class, which are all overlapping, with 5073 objects generated in total. The *Sign* dataset [14] consists of images of 3-classes of road-signs, with a total of 381 objects. The *Sat*

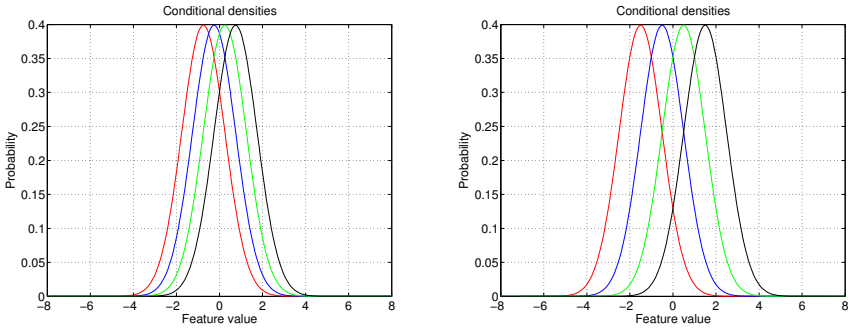


Figure 3.8: Demonstrating the 2nd and 4th experiment in the 4-class case.

dataset [7] consists of 6435 multi-spectral values of a satellite image, with 36 dimensions (4 spectral bands in a 9 pixel neighbourhood). Classes 1, 3, 5 and 6 have been grouped together into a single class, forming a 3-class problem together with classes 2 and 4. The second group of experiments consist of 4-class problems. The *Vehicle* dataset [13] consists of 846 objects of vehicle silhouettes from 4 vehicle types, and the *Digits* dataset consists of examples of ten handwritten digits, originating from Dutch utility maps (available from [13]). In this dataset, Fourier components have been extracted from the original images, resulting in a 76-dimensional representation of each digit. Digits '3', '6', and '9' have been extracted, and the remaining digits grouped into a single class. The experimental methodology involves rotation of the data using a randomised hold-out method in which 80% of the data is used in training, and the remainder for testing, repeated 10 times. Two performance measures are compared, namely the well-known equal-error rate (priors inherent to dataset used), and the simplified VUS measure. Results are compared statistically via a 2-way ANOVA (ANalysis Of VAriance) scheme, with significance judged via a $p-$ value of 0.995. In each experiment, a number of classifiers are compared, with the following abbreviations: *sc* is where unit-variance scaling of the data is used; *pca* is a principal component feature extraction followed by the number of components used; *fisher* and *nlfisher* are the Fisher and non-linear Fisher projections; *nmc*, *ldc*, and *qdc* are nearest-mean, Bayes-linear, and Bayes-quadratic classifiers respectively; *mogc* is a Bayes mixture of Gaussians classifier followed by the number of mixtures used per class; *knn3* is a 3-nearest neighbour classifier; *svc p* is a support vector classifier with a polynomial kernel, followed by the order of the polynomial.

The 3-class table presents the first set of results. The *Banana* dataset shows that the VUS scores tend to track the equal error scores, for example the *nmc* classifier has a high error, and significantly lower VUS than the other classifiers. An interesting result can be seen for the *Sat* case, comparing the second and third models. In this case both classifiers have the same (statistical) error-rate, but significantly different VUS scores (F-value of 275), showing that the

third model is a better choice on average over all operating points. In the *Sign* experiments, similar VUS scores result for all classifiers.

3-class	**Classifier**	**Error**	**VUS**
Banana	sc-pca1 nmc	0.329(0.004)	0.667(0.083)
	sc-mogc2,2,1	0.058(0.003)	0.990(0.002)
	sc-qdc	0.077(0.004)	0.970(0.006)
	sc-ldc	0.091(0.004)	0.964(0.007)
Sat	knn3	0.064(0.002)	0.911(0.020)
	ldc	0.111(0.001)	0.729(0.015)
	qdc	0.108(0.002)	0.862(0.012)
	mogc2,1,2	0.099(0.002)	0.866(0.012)
Sign	sc pca8 svc p2	0.115(0.018)	0.948(0.023)
	sc-pca10 mog2,2,2	0.075(0.003)	0.946(0.025)
	pca5 mog2,2,2	0.099(0.011)	0.954(0.019)
	pca5 qdc	0.179(0.020)	0.945(0.023)

Table 3.5: Experimental results on 3-class problems.

Next the 4-class experiments are considered. A few interesting observations can again be made, for example the first and second classifiers have competing error-rates, but significantly different VUS scores. It appears the *linear* classifier was a far better fit to the data than the *fisher-nmc* model, which only performed well for some operating points. Finally in the *Digits* case, the VUS tended to track the error-rates. It can be seen that some classifiers perform very well, approaching a VUS of 1, whereas others are poor.

4-class	**Classifier**	**Error**	**VUS**
Vehicle	fisher nmc	0.218(0.007)	0.512(0.037)
	ldc	0.219(0.006)	0.714(0.035)
	qdc	0.150(0.010)	0.834(0.036)
	sc-svc p2	0.164(0.007)	0.794(0.022)
	sc-svc p3	0.187(0.009)	0.727(0.039)
	nlfisher qdc	0.208(0.005)	0.724(0.041)
Digits	pca10 mog1,1,1,3	0.119(0.004)	0.985(0.008)
	pca15 mog1,1,1,3	0.114(0.003)	0.955(0.007)
	pca5 mog1,1,1,3	0.133(0.003)	0.956(0.008)
	pca10 qdc	0.127(0.004)	0.978(0.006)
	pca10 ldc	0.211(0.005)	0.704(0.041)
	nlfisher mogc1,1,1,3	0.158(0.003)	0.857(0.024)

Table 3.6: Experimental results on 4-class problems.

The experiments showed the usefulness of the VUS approach in the multi-class case, clearly showing examples where the VUS was required to perform better model selection for classifiers that competed from an equal-error perspective.

3.3.6 Conclusions

This paper considered the extension of the AUC measure to the multiclass case, termed the *volume under the ROC hypersurface*. A simplified extension was considered that evaluates the VUS over the $C-$dimensional ROC surface pertaining to diagonal elements of the confusion matrix only, thus ignoring specific inter- and intra-class performances. This allows for a measure that generalises from the 2-class case, in which high scores result for good classifiers, and low ones for poor ones. It was seen that the VUS bounds vary as a function of the ROC dimensionality, with the lower bound tending to 0 with high dimensionality. A few experiments using known distributions verified the bounds, as well as a proposed numerical integration approach to estimating the hyper-volumes. Finally a set of real experiments were performed that compared equal-errors to VUS scores for a number of competing classifiers. It was found that poor error rates often lead to poor VUS scores, but in some cases competing classifiers in terms of error-rate are not competing in terms of VUS, implying that some classifiers perform better on average over all operating points than others. This work is considered useful to problems involving a low number of classes, restricted by the computational complexity of the ROC generation, but may nevertheless be useful for many real problems.

Acknowledgements: This research is/was supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs, The Netherlands. Special thanks go to Etienne Barnard for his assistance with the analytical bound proof.

Bibliography

[1] N.M. Adams and D.J. Hand. Comparing classifiers when misallocation costs are uncertain. *Pattern Recognition*, 32(7):1139–1147, 1999.

[2] C.B. Barber and H. Huhdanpaa. Qhull. *The Geometry Center, University of Minnesota, http://www.geom.umn.edu/software/qhull*.

[3] A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[4] S. Dreisetl, S. Ohno-Machado, and M. Binder. Comparing trichotomous tests by three-way ROC analysis. *Medical Decision Making*, 20(3):323–331, 2000.

[5] R.P.W. Duin, P. Juszcak, P. Paclik, E. Pekalska, D. de Ridder, and D.M.J. Tax. Prtools, a matlab toolbox for pattern recognition, January 2004.

[6] D.C. Edwards, C.E. Metz, and R.M. Nishikawa. The hypervolume under the ROC hypersurface of "near-guessing" and "near-perfect" observers in N-class classification tasks. *IEEE Transactions on Medical Imaging*, 24(3):293–299, March 2005.

[7] ELENA project. European ESPRIT 5516 project. *Satimage dataset*, 2004.

[8] C. Ferri, J. Hernandez-Orallo, and M.A. Salido. Volume under the roc surface for multi-class problems. *Proc. of 14th European Conference on Machine Learning*, pages 108–120, 2003.

[9] D.J. Hand and R.J. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45:171–186, 2001.

[10] T.C.W. Landgrebe and R.P.W Duin. On Neyman-Pearson optimisation for multiclass classifiers. *Sixteenth Annual Symposium of the Pattern Recognition Assoc. of South Africa*, November 2005.

[11] C. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, 3(4), 1978.

[12] D. Mossman. Three-way roc's. *Medical Decision Making*, 19:78–89, 1999.

[13] P.M. Murphy and D.W. Aha. UCI repository of machine learning databases, ftp://ftp.ics.uci.edu/pub/machine-learning-databases. *University of California, Department of Information and Computer Science*, 1992.

[14] P. Paclík. Building road sign classifiers. *PhD thesis, CTU Prague, Czech Republic*, December 2004.

[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 University NY 10012*, 2001.

[16] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42:203–231, 2001.

3.3.7 APPENDIX: Proof of the simplified lower VUS bound

Figures 3.5 and 3.6, graphically depicted a random 2-class and 3-class ROC plot. We refer to the volume consumed by a random classifier as the lower bound. In the 2-class case, the lower bound can be written as in Equation 3.20, since $\xi_{2,2}$ is the straight line $1 - \xi_{1,1}$.

$$\begin{aligned} AUC_{random} &= \int_0^1 (1 - \xi_{1,1}) d\xi_{1,1} \\ &= \left[\xi_{1,1} - \tfrac{1}{2}\xi_{1,1}^2 \right]_0^1 \\ &= \tfrac{1}{2} \end{aligned} \tag{3.20}$$

In the 3-class case, the volume can be computed analytically by considering the volume under the plane $1 - \xi_{1,1} - \xi_{2,2}$:

$$\begin{aligned} VUS_{random} &= \int_0^1 \int_0^{1-\xi_{1,1}} (1 - \xi_{1,1} - \xi_{2,2}) d\xi_{2,2} d\xi_{1,1} \\ &= \tfrac{1}{2} \int_0^1 (1 - \xi_{1,1})^2 d\xi_{1,1} \\ &= \tfrac{1}{2}\tfrac{1}{3} \\ &= \tfrac{1}{6} \end{aligned} \tag{3.21}$$

Similarly, the 4-class case considers the volume under the hyperplane $1 - \xi_{1,1} - \xi_{2,2} - \xi_{3,3}$:

$$\begin{aligned} VUS_{random} &= \int_0^1 \int_0^{1-\xi_{1,1}} \int_0^{1-\xi_{1,1}-\xi_{2,2}} (1 - \xi_{1,1} - \xi_{2,2} - \xi_{3,3}) d\xi_{3,3} d\xi_{2,2} d\xi_{1,1} \\ &= \tfrac{1}{2} \int_0^1 \int_0^{1-\xi_{1,1}} (1 - \xi_{1,1} - \xi_{2,2})^2 d\xi_{2,2} d\xi_{1,1} \\ &= -\tfrac{1}{2}\tfrac{1}{3} \int_0^1 \left[(1 - \xi_{1,1} - \xi_{2,2})^3 \right]_0^{1-\xi_{1,1}} d\xi_{1,1} \\ &= \tfrac{1}{2}\tfrac{1}{3}\tfrac{1}{4} = \tfrac{1}{24} \end{aligned} \tag{3.22}$$

As C increases, it can be seen that the VUS calculation can be simplified by using the following well-known integration rule recursively:

$$\int (ax + b)^n = \frac{(ax + b)^{n+1}}{a(n + 1)}, n \neq -1 \tag{3.23}$$

The bound for any C can then be computed as follows:

$$
\begin{aligned}
&VUS_{random} = \\
&\int_0^1 \int_0^{1-\xi_{1,1}} \int_0^{1-\xi_{1,1}-\xi_{2,2}} \cdots \int_0^{1-\xi_{1,1}-\xi_{2,2}-\ldots\xi_{C-2,C-2}} (1-\xi_{1,1}-\xi_{2,2}-\ldots\xi_{C-1,C-1}) \\
&d\xi_{C-1,C-1} \ldots d\xi_{2,2} d\xi_{1,1} \\
&= \tfrac{1}{2} \int_0^1 \int_0^{1-\xi_{1,1}} \cdots \int_0^{1-\xi_{1,1}-\xi_{2,2}-\ldots\xi_{C-3,C-3}} (1-\xi_{1,1}-\xi_{2,2}-\ldots\xi_{C-2,C-2})^2 \\
&d\xi_{C-2,C-2} \ldots d\xi_{2,2} d\xi_{1,1} \\
&= \tfrac{1}{2}\tfrac{1}{3} \int_0^1 \int_0^{1-\xi_{1,1}} \cdots \int_0^{1-\xi_{1,1}-\xi_{2,2}-\ldots\xi_{C-4,C-4}} (1-\xi_{1,1}-\xi_{2,2}-\ldots\xi_{C-3,C-3})^3 \\
&d\xi_{C-3,C-3} \ldots d\xi_{2,2} d\xi_{1,1} \\
&= \tfrac{1}{2}\tfrac{1}{3}\tfrac{1}{4} \cdots \tfrac{1}{C} = \tfrac{1}{C!}
\end{aligned}
\tag{3.24}
$$

Chapter 4

Multiclass ROC analysis for large numbers of classes

4.1 Overview

Contributions in chapter 3, as well as recent works such as [4], [2], and [9] have shown that 2-class ROC techniques can indeed be extended to the multiclass case. However these approaches are only practical for problems with low numbers of classes. The first part of this chapter shows how the computational complexity increases exponentially with the number of classes. To cope with this restriction, other approaches have been taken to perform typical ROC tasks without computing the full ROC. For example [5] and [8] present a simplified approach to evaluating multiclass classifiers independently of the operating point. In [6] as well as the first part of this chapter, algorithms are presented that perform cost-sensitive optimisation using a search paradigm. In [3] an approach is taken that attempts to locate a multiclass ROC front using an evolutionary sampling approach.

The second part of this chapter presents a novel approach to estimating the entire multiclass ROC hypersurface. This can often be done efficiently because it was found that in many practical problems, several ROC dimensions are independent, allowing the calculation to be decomposed considerably. The resultant ROC representation can be used subsequently for any type of ROC analysis. An algorithm is presented that efficiently analyses interactions (over various operating points), in order to identify potential simplifications.

The problems and challenges raised by this chapter will in all likelihood be an active area of research in the years to come, but some of these contributions could form a good base. An approach that also forms a good basis for further research is the evolutionary approach proposed in [3]. Sampling the operating characteristic in an efficient way may be another good approach that takes advantage of a lower intrinsic complexity of the multiclass ROC. Recent additions to the multiclass cost sensitive optimisation problem such as [7] and [1] show that there is still scope in this area for further research. Primary drivers here are computational efficiency, and the quest for a global solution.

Bibliography

[1] N. Abe, B. Zadrozny, and J. Langford. An iterative method for multi-class cost-sensitive learning. *Proceedings of the 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 3–11, August 2004.

[2] D.C. Edwards, C.E. Metz, and M.A. Kupinski. Ideal observers and optimal ROC hypersurfaces in N-class classification. *IEEE Transactions on Medical Imaging*, 23(7):891–895, July 2004.

[3] R.M. Everson and J.E. Fieldsend. Multi-class ROC analysis from a multi-objective optimisation perspective. *Pattern Recognition Letters, Special issue on ROC analysis*, 27, 2005.

[4] C. Ferri, J. Hernandez-Orallo, and M.A. Salido. Volume under the roc surface for multi-class problems. *Proc. of 14th European Conference on Machine Learning*, pages 108–120, 2003.

[5] D.J. Hand and R.J. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45:171–186, 2001.

[6] N. Lachiche and P. Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. *Proc. 20th Int. Conf. on Machine Learning, Washington DC*, pages 416–423, 2003.

[7] D.B. O'Brien and R.M. Gray. Improving classification performance by exploring the role of cost matrices in partitioning the estimated class probability space. *Proceedings of the ICML2005 workshop on ROC analysis in machine learning, Bonn, Germany*, 2005.

[8] F. Provost and P. Domingos. Well trained PETs: Improving probability estimation trees. *CeDER Working Paper IS-00-04, Stern School of Business, New York University NY 10012*, 2001.

[9] A. Srinivasan. Note on the location of optimal classifiers in N-dimensional ROC space. *Oxford University Computing Laboratory Technical report PRG-TR-2-99*, October 1999.

4.2 Approximating the multiclass ROC by pairwise analysis

This section has been published as 'Approximating the multiclass ROC by pairwise analysis', by T.C.W. Landgrebe and R.P.W. Duin, in *Pattern Recognition Letters*, **28(13)**:1747-1758, 2007.

Abstract

The use of Receiver Operator Characteristic (ROC) analysis for the sake of model selection and threshold optimisation has become a standard practice for the design of 2-class pattern recognition systems. Advantages include decision boundary adaptation to imbalanced misallocation costs, the ability to fix some classification errors, and performance evaluation in imprecise, ill-defined conditions where costs, or prior probabilities may vary. Extending this to the multiclass case has recently become a topic of interest. The primary challenge involved is the computational complexity, that increases to the power of the number of classes, rendering many problems intractable. In this paper the multiclass ROC is formalised, and the computational complexities exposed. A pairwise approach is proposed that approximates the multi-dimensional operating characteristic by discounting some interactions, resulting in an algorithm that is tractable, and extensible to large numbers of classes. Two additional multiclass optimisation techniques are also proposed that provide a benchmark for the pairwise algorithm. Experiments compare the various approaches in a variety of practical situations, demonstrating the efficacy of the pairwise approach.

4.2.1 Introduction

In pattern recognition, the goal is to choose/optimise representations and classifiers that provide acceptable discriminability between the various classes. Typically one output exists per class, with a new incoming object being assigned to the highest output. The outputs can be weighted[1] in order to vary the trade-offs that exist between classes. These weights are called the operating weights[2].

Once a suitable classifier model has been found, the next step is typically to optimise these operating weights to suit the given problem. For example in the equal/minimum-error rate situation ([4]), e.g. face detection ([21]), errors should be the same for all classes. In detection problems it is often the case that some errors should be fixed, and the others minimised (see e.g. [6], and [14]). In cost-sensitive problems, different classification outcomes have associated costs/penalties ([2]), and so the optimisation problem is to find a set of operating weights that result in the lowest overall loss/risk. Other optimisation scenarios exist, such as the necessity to choose and optimise the best classifier when class priors/costs are unknown (see [22] and [1]), or varying ([15]). At this point we emphasise that of interest in this case is the optimisation of classifier operating weights, and not on internal parameters of the model. Any internal

[1] Even non-probabilistic classifiers e.g. support vector classifiers can be weighted in this manner – outputs in this case are distances to support vectors, which are analogous to probabilities ([16]).

[2] Conceptually similar to the 2-class concept of "thresholds". In the multiclass case, weighting of the output allows for a generalisation of the ROC, with the final discrimination decision based on the highest weighted output

adjustment of the model e.g. varying the thresholds in a one-vs-all multiclass scheme, results in a new model, and a new operating characteristic.

Receiver Operator Characteristic (ROC) analysis ([18], [9]) was developed for the 2-class case, allowing the classifier threshold to be studied for all possible combinations of the two respective classification errors involved, namely the false positive rate, and the false negative rate (with each combination known as the *operating point*). As such, this tool has become standard in the aforementioned optimisation scenarios, but applying it to the multiclass case is more challenging, and poorly understood. Recently some work has begun in this area, such as an extension to the 3-class case in [19], a study of the feasibility of the extension in [23], and investigating some formalisations in [10], and [6]. For the imprecise case in which priors/costs are unknown, the work in [11] presents an approximate method for evaluating classifiers in these conditions, extending the well known Area Under the ROC (AUC) measure ([3]), resulting in the approximate VUS (Volume Under the ROC Surface).

However, even though several works have investigated multi-class operating characteristics, no known methods exist that result in the practical construction of a multiclass ROC that extends to problems involving large numbers of classes, which is necessary in order to perform the optimisations and analyses as discussed earlier. This is attributed primarily to the computational complexity of the analysis. Recently two approaches emerged that are useful for multiclass cost-sensitive optimisation, capable of using input costs and priors to optimise operating weights. The first is a hill-climbing approach as presented in [13] that can easily be adapted to the cost-sensitive case, and the second is an evolutionary approach ([8]), attempting to find a global solution to the minimisation problem. The problem with the former approach is that it is very susceptible to local minima, dealing with interactions between classes sub-optimally. The algorithm optimises the classifier operating weights successively, which is not ideal for arbitrary interactions, but nevertheless improves performance in many cases. The latter approach has only been demonstrated in problems with low numbers of classes. The approach involves sampling of operating points, which becomes less tractable as the number of classes increases, due to an exponential computational complexity (shown later).

In this paper the multiclass ROC approach is formalised, and computational complexity investigated as a function of both the number of classes, and the resolution of the operating characteristic. This shows that computing the multiclass ROC is feasible for low numbers of classes (C), but rapidly becomes intractable as C increases. A new multiclass ROC approach is proposed, in which ROC curves are generated between each type of classification error, characterising the interaction between each pair, but ignoring other interactions. In the cost-sensitive scenario, these characteristics are interrogated to obtain the most optimal operating weight pairs to suit the priors and costs. This involves exhaustively searching pair combinations in order to select the most appropriate weight pairs, followed by a normalisation procedure that "calibrates" weights against each other. The number of weight pairs is reduced for

large C problems by excluding weight pairs exhibiting little/no interaction to reduce computation required. Even though this algorithm is not optimal, it will be shown that it is both computationally tractable, and performs well over many real problems. In addition to the pairwise multiclass ROC method, this paper also presents two simple approaches to the cost-sensitive optimisation problem, consisting of a naive approach that ignores interactions between operating weights, and a greedy-search approach that accounts for interactions (to an extent). These methods provide a benchmark with which to compare the pairwise approach. As a further benchmark, the algorithm in [13] is also included.

The paper is structured as follows: A notational overview and formalism is presented in Section 4.2.2, followed by a formalisation of multiclass ROC construction and an analysis of computational requirements in Section 4.2.3. The naive and greedy multiclass cost-sensitive optimisation algorithms are presented in Section 4.2.4, followed by the proposed all-pairs approach in Section 4.2.5. A variety of experiments are discussed in Section 4.2.6, demonstrating the various algorithms in cost-sensitive scenarios. Conclusions are given in Section 4.2.7.

4.2.2 Notation and formalisation

Consider a C-class problem, with each class denoted $\omega_1, \omega_2, \ldots \omega_C$, and new observations characterised by vector $\mathbf{x}$, with dimensionality d. The class conditional probability of ω_i is denoted $p(\mathbf{x}|\omega_i)$, with prior probability $p(\omega_i)$. Class assignment is based on the highest posterior output, denoted $p(\omega_i|\mathbf{x})$, for the ith class, formalised as:

$$argmax_{i=1}^{C} p(\omega_i|\mathbf{x}) \tag{4.1}$$

The posterior is computed via Bayes formula:

$$p(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)p(\omega_i)}{p(\mathbf{x}|\omega_1)p(\omega_1) + p(\mathbf{x}|\omega_2)p(\omega_2) + \ldots + p(\mathbf{x}|\omega_C)p(\omega_C)} \tag{4.2}$$

A multiclass classifier is evaluated by inspection of a $C \times C$ dimensional confusion rate matrix $\boldsymbol{\xi}$ as defined in Table 4.1. Each element is referenced as $\xi_{i,j}$. In order to compute each confusion $\xi_{i,j}$ (the fraction of ω_i classified as ω_j weighted by priors), the following integration is performed:

$$\xi_{i,j} = p(\omega_i) \int p(\mathbf{x}|\omega_i) I_j(\mathbf{x}) dx \tag{4.3}$$

The indicator function $I_{ij}(\mathbf{x})$ specifies the relevant domain:

$$I_j(\mathbf{x}) = \begin{cases} 1 \text{ if } p(\omega_j|\mathbf{x}) > p(\omega_k|\mathbf{x}) \ \forall k, k \neq j \\ 0 \text{ otherwise} \end{cases} \tag{4.4}$$

Equation 4.3 allows any confusion matrix output to be computed, generalised for both diagonal elements (performances), and off-diagonal elements (errors).

		estimated			
		ω_1	ω_2	$\ldots$	ω_C
	ω_1	$\xi_{1,1}$	$\xi_{1,2}$	$\ldots$	$\xi_{1,C}$
true	ω_2	$\xi_{2,1}$	$\xi_{2,2}$	$\ldots$	$\xi_{2,C}$
	$\vdots$	$\vdots$		$\ddots$	
	ω_C	$\xi_{C,1}$	$\xi_{C,2}$	$\ldots$	$\xi_{C,C}$

Table 4.1: Defining the multi-class confusion rate matrix $\boldsymbol{\xi}$.

In the practical case where distributions are unknown, and only representative examples per class are available, a confusion matrix $\boldsymbol{cm}$ is constructed, generated via application of a representative independent test set. These $\boldsymbol{cm}$ outputs are normalised by the absolute number of objects N_i per class ω_i, $\boldsymbol{N} = [N_1, N_2, \ldots N_C]^T$, resulting in the confusion rate matrix, where each element $\xi_{i,j} = \frac{cm_{i,j}}{N(i)}$.

Each posterior output $p(\omega_i|\mathbf{x})$ can be weighted by the scalar $\phi_i, \phi_i \geq 0$, in order to control trade-offs between the various classification errors (note that all classifier outputs are scaled between $[0, 1]$, irrespective of the classifier type). The classifier weight vector $\Phi = [\phi_1, \phi_2, \ldots \phi_C]$ is thus the mechanism for manipulating a classifier's decision boundary. Note that there are $C - 1$ degrees of freedom, and thus in the two-class case, $\Phi = [\phi_1, (1 - \phi_1)]$, since there is only one degree of freedom. Similarly, where $C > 2$, one of the weights is generally set to an arbitrary positive value e.g. for a 5-class problem $\Phi = [1, \phi_2, \phi_3, \phi_4, \phi_5]$. Class assignment can thus be modified as:

$$argmax_{i=1}^{C} \phi_i p(\omega_i|\mathbf{x}) \tag{4.5}$$

In the cost sensitive case, the optimisation goal is to minimise the various classification errors, using a framework in which the importance of each error is weighted via a classification *cost*, and the respective prior probability. The cost sensitive situation is typically evaluated by considering the overall system *loss*, L, given a matrix of costs $\boldsymbol{s}$ (see Table 4.2), and prior probabilities. Now

	ω_1	ω_2	$\ldots$	ω_C
ω_1	$s_{1,1}$	$s_{1,2}$	$\ldots$	$s_{1,C}$
ω_2	$s_{2,1}$	$s_{2,2}$	$\ldots$	$s_{2,C}$
$\vdots$	$\vdots$		$\ddots$	
ω_C	$s_{C,1}$	$s_{C,2}$	$\ldots$	$s_{C,C}$

Table 4.2: Defining the multi-class cost matrix $\boldsymbol{s}$.

the loss can be defined (in this paper the profits (diagonals) are set to zero):

$$L = \sum_{i=1}^{C} p(\omega_i)(\sum_{j=1, i \neq j}^{C} \xi_{i,j} s_{i,j}) - \sum_{i=1}^{C} p(\omega_i) \xi_{i,i} s_{i,i} \tag{4.6}$$

In the left of Figure 4.1, a 4-class example is shown for two different operating points, consisting of Gaussian-distributed classes $\omega_1, \omega_2, \ldots, \omega_4$, with

means occurring at $\mu_1 = -3, \mu_2 = 0, \mu_3 = 3, \mu_4 = 6$ respectively, with unit variance. The left plot shows an operating point involving equal priors and $\Phi = [1.0, 1.0, 1.0, 1.0]$, and the right plot shows another operating point $\Phi = [1.0, 0.3, 1.0, 1.0]$, in which $\xi_{1,2}$ decreases at the expense of $\xi_{2,1}$.

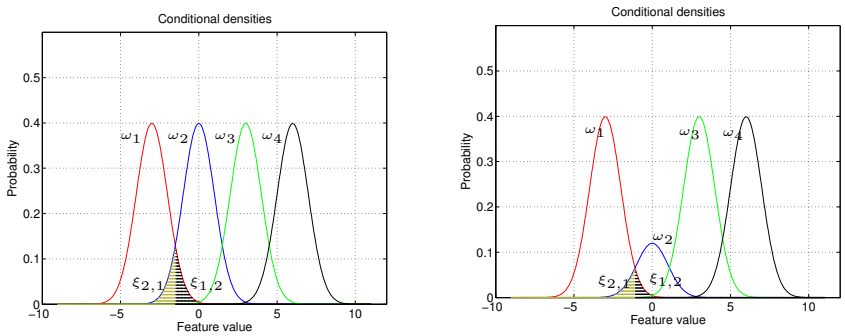


Figure 4.1: Probability density functions for the 4-class example with known distributions. Two operating points are shown, with the left plot involving equal output weighting, and the right with a higher ϕ_2 and lower ϕ_1.

4.2.3 Multiclass ROC

Implementation

Referring to Figure 4.1, the plots show only two operating points, corresponding to two different operating weight settings. In fact, any combination of weightings results in a different operating point. The challenge in multiclass optimisation is in understanding the relation between a weight modification and the corresponding alteration of the confusion matrix, which depicts the consequences of the new operating point. Multiclass operating weight (analogous to threshold) optimisation is thus the process by which the optimal set of weights Φ^* is found to suit the problem at hand. Note that there may be multiple optimal solutions for $C > 2$. Multiclass ROC analysis involves the generation of a hypersurface consisting of all possible combinations of Φ, allowing these various possible confusion matrices to be characterised. The dimensionality of the hypersurface is $C^2 - C$ (diagonal elements are superfluous), which can be constructed by adapting Equation 4.3. In this case, each output between class i and j is weighted by ϕ_i as follows:

$$\xi_{i,j}(\Phi) = \phi_i p(\omega_i) \int p(\mathbf{x}|\omega_i) I_j(\mathbf{x}|\Phi) dx \tag{4.7}$$

The indicator function $I_{ij}(\mathbf{x}|\Phi)$ is as in 4.4, except each posterior is multiplied by the corresponding class weight:

$$I_j(\mathbf{x}|\Phi) = \begin{cases} 1 \text{ if } \phi_j p(\omega_j|\mathbf{x}) > \phi_k p(\omega_k|\mathbf{x}) \ \forall k, \\ \ k = 1, 2, \ldots C, \ k \neq j \\ 0 \text{ otherwise} \end{cases} \tag{4.8}$$

Since the classifier has $(C-1)$ degrees of freedom, the ROC is constructed by generating a $(C-1)$ dimensional grid of all possible operating weights $\mathbf{\Phi}$, with resolution r. This results in a set of confusion rate matrices, corresponding to each weight combination, denoted $\boldsymbol{\xi}(\mathbf{\Phi})$. Different elements of the confusion matrices as a function of Φ are the dimensions of the ROC. An important consideration for practical implementation is the choice of the resolution r and scale Θ of each weight. The resolution must be fine enough, and the scale adequately chosen to ensure the operating characteristic is well sampled. Experiments in this paper consider a logarithmic scale, sampled 80 times, with $10^{-3} \leq \Theta \leq 10^{3}$.

The full multiclass ROC has been computed for the analytic example (see Figure 4.1), of which 3 of the 12 $(4 \times 4 - 4)$ ROC dimensions are plotted in Figure 4.2, illustrating the respective interactions. The full ROC in this case results in $80^{4-1} = 512 \times 10^{3}$ operating points.

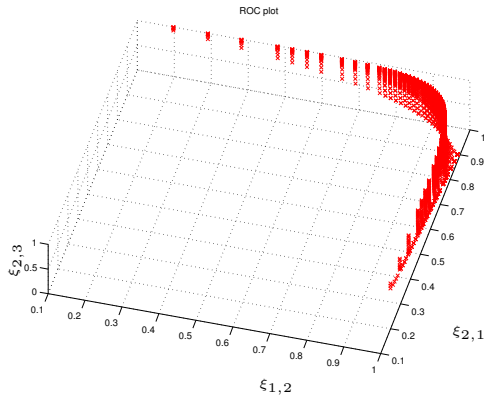


Figure 4.2: Illustrating the ROC dimensions $\xi_{1,2}$, $\xi_{2,1}$, and $\xi_{2,3}$ for the example.

Computational considerations

In the 2-class case, ROC construction involves generating a 1-dimensional grid of weights (since there are $C-1$ weights, there is only 1 DOF in this case), with r steps across the output range. In the multiclass case $(C > 2)$, computing the ROC involves the generation of a $C-1$ dimensional grid with r steps, resulting in r^{C-1} operating points. This increase to the power of the number of classes minus 1 $(O(r^{C-1}))$ explodes the computational complexity with increasing C, becoming infeasible to compute for all but low C problems. To illustrate the severity of this problem, Figure 4.3 plots the number of calculations (and number of memory slots for storage of the hypersurface) as a function of the ROC resolution r, for a number of C-values. It is clear that for high C, computational complexity becomes prohibitive. Use of a lower r does reduce the computation required, but this is at the expense of poorer sampling,

which could lead to a poorly sampled ROC surface[3]. To illustrate, consider the following real problems:

- *Satellite* (obtained from [7]), which consists of 6 classes of multi-spectral remote sensing data. An ROC resolution of 80 requires $80^{6-1} = 3.28 \times 10^9$ calculations.
- *Letter* (obtained from [20]) consists of 26 different classes of hand-written digits, requiring $80^{26-1} = 3.78 \times 10^{47}$ calculations for $r = 80$, which is clearly prohibitive.

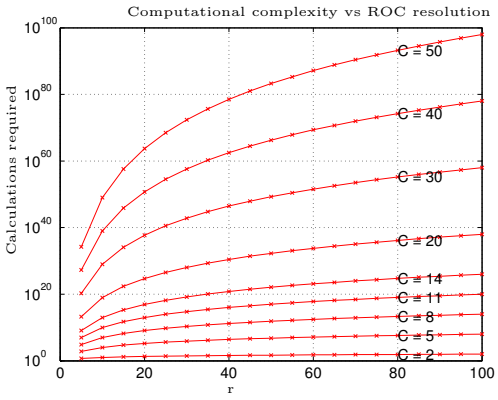


Figure 4.3: The relationship between computational complexity, ROC resolution, and C.

Thus even though it is theoretically possible to construct any multiclass ROC, only problems with low C are feasible to compute. This is the justification for seeking approximate techniques for performing multiclass ROC analysis.

4.2.4 Naive and greedy cost-sensitive optimisation algorithms

In the cost-sensitive case, obtaining a set of operating weights to suit new costs $\boldsymbol{s}$ and priors $\boldsymbol{p}$ can be viewed as an optimisation problem ([8]). In this Section two simple algorithms are presented that are suitable for this task. We stress that these approaches simply result in a single set of weights, and not an operating characteristic, and are thus unable to fulfil several functions that are made possible by ROC analysis. However, they are still useful and relatively straightforward in the cost-sensitive scenario, allowing for an adaptation of an

[3]Different classifier models e.g. a support vector classifier or a Bayes linear classifier, typically result in outputs (e.g. posteriors) with different scales, and thus choosing a scale suitable to the classifier used could improve the computational situation by allowing for a lower r.

unoptimised classifier (the "default" classifier). These algorithms also provide a basis for comparison with respect to the pairwise ROC approach.

The objective of both algorithms is to obtain the most appropriate weight vector Φ^*, achieved by searching for a solution that reduces an initial system loss (using Equation 4.6) by varying individual classifier weights. These approaches cannot guarantee a global minimum, but should always improve on an initial "default" classifier that was trained to a typically equal-error operating point (and accounting for class priors). This initial classifier results in a baseline confusion rate matrix used to obtain the initial loss for a given set of conditions (equivalent to operating weights set to unity).

Naive multiclass cost-sensitive optimisation

The *Naive* algorithm attempts to optimise operating weights to given cost $\boldsymbol{s}$ and prior probability $\boldsymbol{p}$ conditions by varying each operating weight successively and independently, and inspecting overall system loss (Equation 4.6). This approach ignores interaction between different weights, and simply "optimises" each one in turn.

The following steps are taken: A threshold vector Θ is computed, $10^{-3} \leq \Theta \leq 10^3$, with resolution r, using a logarithmic scale. In the first step the first weight ϕ_1 is optimised, resulting in ϕ_1^*. This is obtained by varying ϕ_1 according to Θ, while other weights are fixed, and computing the loss. The minimum loss for this weight is denoted $L^*(\phi_1|\phi_i = 1, \forall i, i \neq 1)$, which is then used to obtain ϕ_1^*, computed using Equation 4.6:

$$L^*(\phi_1|\phi_i = 1, \forall i, i \neq 1) \quad = \quad min(L(\phi_1 = \Theta)), \quad \phi_1^* \quad = \quad \Theta(L_1^*) \tag{4.9}$$

This "optimises" the first weight independently of the others. The same procedure is then followed for all other weights, resulting in an "optimal" weight value in each case, ignoring all interactions:

$$\begin{array}{rcrcl} L^*(\phi_2|\phi_i = 1, \forall i, i \neq 2) & = & min(L(\phi_2 = \Theta)), \phi_2^* & = & \Theta(L_2^*) \\ L^*(\phi_3|\phi_i = 1, \forall i, i \neq 3) & = & min(L(\phi_3 = \Theta)), \phi_3^* & = & \Theta(L_3^*) \\ & \vdots & & & \\ L^*(\phi_k|\phi_i = 1, \forall i, i \neq k) & = & min(L(\phi_k = \Theta)), \phi_k^* & = & \Theta(L_k^*) \\ & \vdots & & & \\ L^*(\phi_C|\phi_i = 1, \forall i, i \neq C) & = & min(L(\phi_C = \Theta)), \phi_C^* & = & \Theta(L_C^*) \end{array} \tag{4.10}$$

Even though the *Naive* algorithm is sub-optimal, it has a computational complexity of $O(rC)$ (where r is the resolution of Θ), extending linearly with C, and is thus scalable to high C problems. The experiments (Section 4.2.6) show that this approach is generally better than an un-optimised approach, but is usually outperformed by other more sophisticated algorithms that do account for interactions.

Greedy multiclass cost-sensitive optimisation

The *Greedy* multiclass optimisation algorithm is quite similar to the *Naive* approach, except that some degree of interaction between weights is accounted for. This is achieved by searching for a (local) optimal set of operating weights by optimising weights with respect to each other in a greedy manner. This involves randomly selecting a weight to update, followed by optimisation relative to other weights. Subsequently another operating weight is randomly selected, and optimised while taking into account previously optimised weights. This process is repeated until all weights are accounted for. In an attempt to avoid local minima, the algorithm is typically run a number of times with different random initialisations. The algorithm is as follows: A threshold vector Θ is computed as in the *Naive* case, and the weight vector Φ is randomly ordered, resulting in Φ^R, with the ith weight denoted ϕ_i^R. The first weight ϕ_1^R is optimised, resulting in ϕ_1^{R*}, by considering all possible Θ, while other weights are fixed, and computing the loss (Equation 4.6). The minimum loss for this weight is denoted $L^*(\phi_1^R|\phi_i^R = 1, \forall i, i \neq 1)$, which is then used to obtain ϕ_1^{R*}, computed using Equation 4.6:

$$L_1^*(\phi_1^R|\phi_i^R = 1, \forall i, i > 1) \quad = \quad min(L(\phi_1^R = \Theta)), \quad \phi_1^{R*} \quad = \quad \Theta(L_1^*) \tag{4.11}$$

The *Greedy* algorithm then proceeds to optimise other weights in the order as per Φ^R. The primary difference to the *Naive* algorithm is that weights are now updated *dependent* on previously updated weights, as follows:

$$\begin{aligned}
L^*(\phi_2^R|\phi_1^R = \phi_1^{R*}, \phi_i^R = 1, \forall i, i > 2) &= min(L(\phi_2^R = \Theta)), \phi_2^{R*} \\
&= \Theta(L_2^*) \\
L^*(\phi_3^R|\phi_1^R = \phi_1^{R*}, \phi_2^R = \phi_2^{R*}, \phi_i^R = 1, \forall i, i > 3) &= min(L(\phi_3^R = \Theta)), \phi_3^{R*} \\
&= \Theta(L_3^*) \\
&\vdots \\
L^*(\phi_k^R|\phi_j^R = \phi_j^{R*}, \forall j, j < k, \phi_i^R = 1, \forall i, i > k) &= min(L(\phi_k^R = \Theta)), \phi_k^{R*} \\
&= \Theta(L_k^*) \\
&\vdots \\
L^*(\phi_C^R|\phi_j^R = \phi_j^{R*}, \forall j, j < C) &= min(L(\phi_C^R = \Theta)), \phi_C^{R*} \\
&= \Theta(L_C^*)
\end{aligned} \tag{4.12}$$

The *Greedy* algorithm has a computational complexity of $O(N_r rC)$, where N_r is the number of algorithm repetitions. This algorithm is thus also extensible to large C. A similar algorithm was also recently proposed in [8], using a more sophisticated search approach.

4.2.5 Pairwise multiclass ROC analysis

The pairwise ROC algorithm investigates interactions between each pair of operating weights by analysing ROC plots between each pair. Only the respective pair operating weights are varied (with other weights held constant), and the

resultant confusion rate matrices stored. For a given problem, the pairs that are most suitable are then chosen to be used. This results in a much more efficient algorithm than the full multiclass ROC, since only individual pairings are considered. The algorithm is simplified further by discounting pairs which are approximately separable based on the Area under the ROC (AUC) criterion. The limitation of this pairwise approach is that the pairs discount interactions not included in the pair, leading to sub-optimality, but we argue that accounting for the most important interactions may result in a good approximation. In an attempt to cater for some degree of further interaction, the algorithm is followed by a post-processing step, using the *Greedy* algorithm in Section 4.2.4.

Given a C-class problem, the algorithm proceeds as follows: In the first step a vector of weight pair indices PI is constructed, consisting of $\frac{C^2-C}{2}$ pairs:

$$\begin{aligned} PI &= [[\phi_1, \phi_2], [\phi_1, \phi_3], [\phi_1, \phi_4], \ldots, [\phi_1, \phi_C], [\phi_2, \phi_3], [\phi_2, \phi_4], \ldots, [\phi_2, \phi_C], \\ &\quad \ldots, [\phi_{C-1}, \phi_C]] \end{aligned} \tag{4.13}$$

The next step involves computing an ROC curve corresponding to each pair $[\phi_i, \phi_j], j > i$, denoted $ROC(\phi_i, \phi_j)$. This is performed via Equation 4.7, varying weights ϕ_i and ϕ_j only, and weights $\phi_k = 1, k = 1, 2, \ldots C, k \neq i, j$. This process results in $\frac{C^2-C}{2}$ ROC plots that can be analysed or interrogated to suit a given problem. In the full multiclass case, the operating characteristic can be analysed directly, but in this case, a secondary step is required to amalgamate information from the most relevant ROC pairs.

Consider for example the cost-sensitive case given a cost $\boldsymbol{s}$ and prior $\boldsymbol{p}$. The task is to select the best ROC pairs to suit the new situation, but there is an ambiguity in the pairwise case, since the same operating weight is optimised $(C-1)$ times e.g. $ROC(\phi_2, \phi_3)$ and $ROC(\phi_2, \phi_4)$ both result in an optimised ϕ_2. The pairwise algorithm selects the pair best suited to the problem by considering all possible pairings. This is sub-optimal because interactions between other classifier weights not in the pair are now ignored, but in some problems certain interactions may be more significant than others. In addition, some pairs may involve more costly implications than others, in which case these pairs should be favoured. Thus the philosophy of the pairwise algorithm is to consider both the degree of interaction, and the severity of an interclass error, resulting in the most optimal pair selection for the given scenario. This is practically achieved by considering all feasible combinations of pairs of results, and selecting the combination with the lowest overall loss. The number of possible unique combinations of pairings is denoted N_{PC}, computed as follows (not to be confused with the total number of pairs):

$$N_{PC} = (C-1)(C-3)(C-5)\ldots 1 \tag{4.14}$$

For example, in the 6-class case, the following PI results ($N_{PC} = (6-1)(6-3)(6-5) = 15$ in this case):

$$\begin{aligned} PI \quad = \quad & [[\phi_1,\phi_2],[\phi_1,\phi_3],[\phi_1,\phi_4],[\phi_1,\phi_5],[\phi_1,\phi_6],[\phi_2,\phi_3],[\phi_2,\phi_4],[\phi_2,\phi_5], \\ & [\phi_2,\phi_6],[\phi_3,\phi_4],[\phi_3,\phi_5],[\phi_3,\phi_6],[\phi_4,\phi_5],[\phi_4,\phi_6],[\phi_5,\phi_6]] \end{aligned} \tag{4.15}$$

Next, a matrix PI_c is constructed from the PI pairs. Each row consists of a different complete set of class pairs covering all classes. For example, in the 6-class case, PI_c is:

$$\begin{bmatrix} PI(1) & PI(10) & PI(15) \\ PI(1) & PI(11) & PI(14) \\ PI(1) & PI(12) & PI(13) \\ PI(2) & PI(7) & PI(15) \\ PI(2) & PI(8) & PI(14) \\ PI(2) & PI(9) & PI(13) \\ PI(3) & PI(6) & PI(15) \\ PI(3) & PI(8) & PI(12) \\ PI(3) & PI(9) & PI(11) \\ PI(4) & PI(6) & PI(14) \\ PI(4) & PI(7) & PI(12) \\ PI(4) & PI(9) & PI(10) \\ PI(5) & PI(6) & PI(13) \\ PI(5) & PI(7) & PI(11) \\ PI(5) & PI(8) & PI(10) \end{bmatrix} \tag{4.16}$$

In the odd case, one weight will not be included, since it cannot be paired. This weight is assigned a finite positive value (e.g. 1), and is excluded from the optimisation, chosen based on lack of importance (e.g. low cost, or low degree of interaction).

Now we have obtained a set of pairwise ROC curves $ROC(\phi_i,\phi_j), i = 1,2,\ldots C, j > i$, and the PI_c matrix, which indicates how the ROC curves can be interrogated based on feasible pairings. We now focus on the cost-sensitive case to illustrate how this ROC approximation can be used. Each pair is optimised using the given $\boldsymbol{s}$ and $\boldsymbol{p}$, resulting in pairs of optimal weights. These values are stored in a matrix with the same size as PI according to the PI pairing, denoted PI^{Φ}. In the next step, all the possible optimised weight pairings are assembled according to PI_c, and re-ordered to correspond with the operating weight ordering, denoted Φ^M.

At this stage an additional step is included to "calibrate" operating weight pairs with respect to other pairs. This is necessary because the pairwise weight optimisation considers the loss associated with the two respective weights, resulting in an optimal weighting between them, but the overall weighting may differ in scale with other groups. Thus it is important to adjust/"calibrate" the various weight pairs with respect to each other. Importantly, the relative values in a weight pair are held constant, but the overall weighting is adjusted in a greedy fashion using an approach similar to the *Greedy* algorithm, with a computational complexity of $O(\frac{r}{2}(C^2-C))$. The "calibrated" weight pairs are denoted Φ^{Mc}.

Subsequent to the computation of Φ^{Mc}, the best weight vector entry is chosen (each row in Φ^{Mc} is one possible operating weighting) by computing the overall system loss according to Equation 4.6 for each entry, and choosing the weight vector resulting in the minimum loss, denoted Φ^*. A final optional

post-processing step attempts to overcome the assumption of pairwise interactions only by adjusting Φ^* according to the *Greedy* algorithm in Section 4.2.4, resulting in Φ^{**}.

Computationally, the pairwise algorithm increases in complexity as follows: Firstly ($\frac{1}{2}(C^2 - C)$) ROC pairs must be computed, each of which have a complexity $O(r)$, resulting in an $O(\frac{r}{2}(C^2 - C))$ calculation, which is tractable even for large C. The next step is to compute all possible pairings, resulting in N_{PC} pairs according to Equation 4.14, followed by the "calibration" step, resulting in an $O(\frac{rN_{PC}}{2}(C^2 - C))$. While problems with lower C are quite tractable (e.g. $N_{PC} = 48$ for $C = 7$, and $N_{PC} = 105$ for $C = 8$), larger C problems become less so (e.g. $N_{PC} = 10395$ for $C = 12$). A useful approach to reduce the computational complexity for large C problems is to remove pairs that are considered unimportant. Fewer pairs implies a smaller N_{PC}, reducing the complexity radically for high C problems. These pairs are chosen based on the degree of interaction - little or no interaction implies that these pairs will play an insignificant role in the optimisation process. The AUC criterion is applied here $AUC(\phi_i, \phi_j) = 1 - \int(\xi_{i,j})d\xi_{j,i}$, measuring the degree of separability for each pair, with high AUC values implying a low degree of interaction. A threshold t_p is used to eliminate these pairs.

In summary, the pairwise algorithm proceeds as follows (for the cost-sensitive case):

- Weight pair indices NI are computed according to Equation 4.13, resulting in $\frac{C^2-C}{2}$ pairs.
- ROC curves are produced for each pair in NI.
- For large C problems, the AUC criterion is applied to each ROC pair, eliminating pairs demonstrating little interaction according to a chosen threshold t_p.
- All possible pairwise combinations PI_c are calculated, resulting in N_{PC} possibilities (or fewer if pairs were removed in the previous step).
- Given a new cost/prior situation, each ROC is optimised, resulting in a set of optimised weight pairs.
- Optimised weight pairs are assembled and ordered to create Φ^M, according to PI_c.
- Each weight pair is "calibrated" to normalise the overall scaling of pair groups with each other, resulting in Φ^{Mc}.
- Each candidate weight vector in Φ^{Mc} is used to compute the overall loss according to Equation 4.6, with best solution chosen as Φ^*.
- An optional *Greedy* optimisation (Section 4.2.4) is applied to Φ^*, resulting in Φ^{**}, attempting to overcome the limitation of the pairwise assumption.

Pairwise optimisation example

An example of the cost-sensitive pairwise ROC analysis approach is discussed, referring to the synthetic problem in the left of Figure 4.4 (called the *PRTools8* dataset), generated by the PRTools pattern recognition toolbox ([5]), consisting of 8 classes, with balanced priors. In this experiment, a Bayes quadratic discriminant has been trained on 500 independent training examples, as depicted in the left of Figure 4.4, showing the resultant default decision boundary on an independent test set with 500 examples. In this equal-cost, equal-prior case, it can be seen that the classifier attempts to maintain the equal error state. A cost-sensitive scenario is considered, in which balanced priors are present, but the following cost matrix $\boldsymbol{s}$ occurs:

```
0.000 0.003 0.113 0.115 0.169 0.133 0.113 0.101
0.044 0.000 0.101 0.140 0.007 0.189 0.183 0.084
0.063 0.136 0.000 0.102 0.061 0.045 0.092 0.159
0.090 0.273 0.235 0.000 0.078 0.016 0.113 0.002
0.023 0.013 0.049 0.048 0.000 0.048 0.049 0.009
0.114 0.091 0.054 0.081 0.029 0.000 0.016 0.090
0.002 0.059 0.057 0.037 0.000 0.006 0.000 0.041
0.030 0.047 0.089 0.068 0.102 0.060 0.087 0.000
```

The default classifier results in a loss of 0.9359 in this example. The pairwise ROC algorithm attempts to reduce this by finding a new set of operating weights, resulting in a loss of 0.6250 (with no post-processing), which is an improvement over the default case. The following operating weights result:

$$\Phi^* = [0.1000, 0.9000, 0.7875, 0.2125, 0.9250, 0.0750, 0.7875, 0.2125] \quad (4.17)$$

These operating weights are applied to the trained classifier, resulting in the decision boundary depicted in the right plot. For comparative purposes, the *Naive* algorithm resulted in a loss of 0.8081 which is better than the default case, but worse than the pairwise algorithm and the *Greedy* algorithm resulted in a loss of 0.6347, which is similar to the pairwise algorithm result.

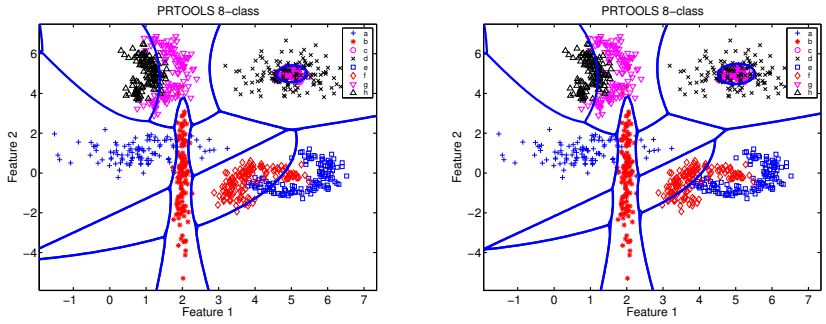


Figure 4.4: Comparing decision boundaries for an 8-class problem in the balanced prior/cost case (left), and in the imbalanced case optimised using the pairwise algorithm in the right plot.

4.2.6 Experiments

A number of experiments are undertaken over a variety of datasets in order to assess the applicability and efficacy of the pairwise multiclass ROC approach, as well as the *Naive* and *Greedy* approaches in the cost-sensitive scenario. The hill-climbing algorithm from [13] is also compared, called *Lachiche*. Two sets of experiments are performed, consisting of *synthetic* and *real* datasets. The *synthetic* experiments analyse performance of the various algorithms in synthetic scenarios suited to the pairwise algorithm, and also investigate the impact of a growing number of classes. The *real* experiments compare performance in realistic scenarios. A trained classifier which is unoptimised (the "default" classifier) is used as the basis for the comparison. The various problems are studied over many different misallocation cost scenarios in order to assess the strengths, weaknesses, and generality of the various algorithms over many situations.

Methodology

Each dataset is analysed, and an appropriate classifier trained (independently), and tested on an independent test set. The chosen classifiers are not necessarily optimal, since the objective is to demonstrate how a classifier can be tuned to a new operating point, rather than how to design a classifier.

The experimental methodology involves the generation of 50 different random cost-matrices (sampled uniformly, and zero diagonal elements), with priors held constant, and comparing the results of the various multiclass cost-sensitive optimisation algorithms. Each different cost-matrix is a new scenario, weighting both inter- and intra-class errors in a variety of different combinations, with the experimental objective of obtaining a diverse set of scenarios with which to fairly compare different algorithms. The results of the various algorithms are benchmarked against the default unoptimised classifier, with the loss measure used to evaluate performance (Equation 4.6). For each new set of costs, the following algorithms are compared:

- *Default* algorithm, in which no optimisation occurs, and the overall loss is simply computed given the default confusion matrix.
- *Naive* algorithm, as described in Section 4.2.4.
- *Greedy* algorithm, as described in Section 4.2.4, with $N_r = 3$.
- *Lachiche* algorithm, implemented according to [13].
- *Pairs* algorithm, which is the multiclass pairwise algorithm as described in Section 4.2.5.
- *PairsG* algorithm, which is the same as the *Pairs* algorithm, but the post-processing *Greedy* optimisation step is included, using the first stage to initialise the search, attempting to overcome the pairwise limitation.

Even though loss (or cost) is a useful measure to observe relative differences between algorithms, it is dependent on the number of classes, and the priors, and scales according to the costs. This makes it difficult to assess the absolute benefit of an optimisation, and how well the classifier is performing for a given set of costs and priors. In [17], the L measure is rescaled using $\boldsymbol{s}$ and $\boldsymbol{p}$ into the Mean Subjective Utility Score (MSU), that overcomes these issues, resulting in a performance measure (higher scores imply improvement) that scales between 0 and 1:

$$\begin{aligned} MSU &= \sum_{i=1}^{C}(\sum_{j=1,i\neq j}^{C}\xi_{i,j}s_{ij}^{'}) \\ S^{'} &= -\beta_1 S + \beta_2 \end{aligned} \tag{4.18}$$

$$\begin{aligned} -\beta_1(\sum_{i=1}^{C}p(\omega_i)s_{i,i}) + \beta_2 &= 1 \\ -\beta_1(\tfrac{1}{C}\sum_{i=1}^{C}p(\omega_i)\sum_{j=1,j\neq i}^{C}s_{i,j}) + \beta_2 &= 0 \end{aligned} \tag{4.19}$$

This measure can now be used to gauge the percentage improvement that an optimisation may have, instead of using an arbitrary scale. For example, in Section 4.2.5, the *Default* classifier resulted in a loss of 0.9359, and the pairwise optimisation reduced this to 0.6250. The degree of improvement is not known, but the MSU scores in this case are 0.8016 and 0.8675 respectively, implying that the pairwise approach has improved performance by 6.59%. Thus the MSU score is chosen to compare results since we can immediately assess the impact an optimisation has on performance.

In summary, for each experiment, 50 different cost-sensitive scenarios are considered for the various approaches, each of which results in a new operating weight vector. This is used to weight the classifier outputs, resulting in a new confusion matrix based on the independent test set. Each approach is compared to the default unoptimised classifier, with the MSU measure showing the percentage improvement. The experiments consider the overall improvement the various approaches give over the various scenarios. Note that if an optimisation results in a performance worse than the default case, the improvement in performance is considered to be zero.

Dataset descriptions

The various algorithms are evaluated by considering a wide variety of problems. In Table 4.3, the experimental datasets are summarised in terms of training/test sizes, numbers of classes, and dimensionality. Also included are the number of pairs involved in each case, followed by the AUC threshold t_p used (if any), which reduces the numbers of pairs used, shown in the 'Pairs2' column. The table then shows which classifier has been chosen in each case, as well as the mean classification error on the default classifier $\epsilon = mean(\sum_{i=1}^{C}\sum_{j=1}^{C}\xi_{i,j}, i \neq j)$ based on the test set. The classifier 'qdc' is a Bayes quadratic classifier, 'mogc' is a Bayes mixture of Gaussians classifier followed by the number of mixtures, 'pca' is a principal components feature extraction, followed by the number of components used, and 'fisherm' is a Fisher projection on the original data. The first two entries are the *synthetic* datasets, with the remainder consisting

of *real* datasets. The *PRTools8*, *Letter*, and *Satellite* datasets have been introduced earlier. The *PRTools16* dataset is a second synthetic dataset based on the *PRTools8* dataset, with a repetition of the 8-classes in feature space, resulting in 16 classes (simply by adding a value of 10 to both features). The *CBands* dataset consists of chromosome band profiles [12]. The *Digits* dataset consists of examples of ten handwritten digits, originating from Dutch utility maps (available from [20]). In this dataset, Fourier components have been extracted from the original images, resulting in a 76-dimensional representation of each digit. Comparing t_p for the *Letter* and *Cbands*, it can be seen that the *Cbands* appears to contain more interactions, since more pairs result even with a lower threshold, suggesting that there is a higher "intrinsic complexity" than in the *Letter* case.

Dataset	Train/Test	C	d	Pairs	t_p	Pairs2	Classifier	ϵ
PRTools8	500/500	8	2	28	-	28	qdc	13.05%
PRTools16	1000/1000	16	2	120	0.996	17	qdc	13.05%
Letter	16000/4000	26	16	325	0.980	22	fisherm qdc	12.50%
Cbands	6000/6000	24	30	276	0.975	34	pca10 mogc2	17.63%
Digits	1000/1000	10	76	45	0.980	10	ldc	20.50%
Satellite	4435/2000	6	17	15	-	15	fisherm mogc2	12.50%

Table 4.3: Important dataset statistics, showing number of train/test objects, the number of classes, dimensions, number of pairwise ROC curves required, the pairwise AUC threshold t_p, followed by the actual number of pairs used ('Pairs2'), the classifier model, and the mean classification error ϵ of the unoptimised classifier.

Computationally, the pairwise algorithms for the *PRTools16*, *Letter*, *CBands*, and *Digits* datasets used a reduced number of pairs by eliminating the least interacting pairs. This reduced the N_{PC} in each case, radically reducing the computation required, since this becomes the determining factor with respect to computational complexity when C is large.

Results

In Table 4.4, the results of the various algorithms are summarised over the 50 different cost-sensitive scenarios for each dataset. Each result subtracts the MSU score of the respective algorithm with the MSU result (%/100) of the default unoptimised classifier. Higher results thus indicate larger improvements by the respective algorithm. The Table attempts to result in an overall analysis of the use of each algorithm, showing the minimum, median, maximum, best, and mean performance over the 50 different conditions.

The *PairsG* and *Greedy* algorithms are the two most promising approaches, which are compared directly via Table 4.5. These results show the number of experiments (out of 50) that the *PairsG* algorithm was superior.

In order to illustrate the experimental process in detail, the full results of the *Letter* dataset are shown in Figure 4.5, comparing the various MSU results for 50 different cost-sensitive scenarios. It can be seen that the default unoptimised classifier is consistently inferior. The *Naive* algorithm is consistently

better than the default classifier, but is inferior to the *Greedy* and *PairsG* algorithms. The *PairsG* algorithm is superior in most cases, suggesting that this is generally the best approach. The *Pairs* algorithm (omitted) did not work well by itself, suggesting higher order (i.e. > 2) interactions. However, the superior performance of the *PairsG* algorithm shows that the *Pairs* algorithm did provide a good starting point for the post-processing, by considering the most important pairs. The *Lachiche* results are omitted because no improvements resulted under any conditions.

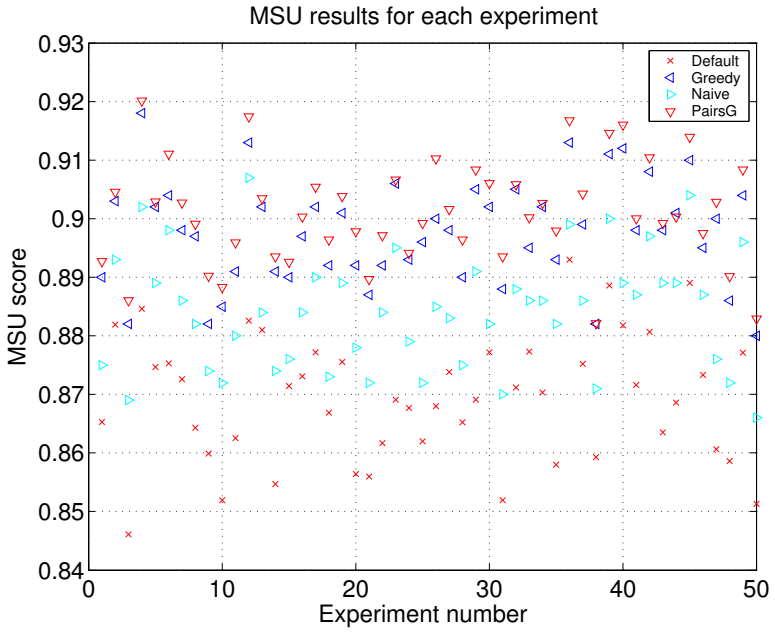


Figure 4.5: Detailed results for the *Letter* dataset, comparing MSU performance (larger scores are better) across 50 different experiments, involving randomly varying cost specifications.

Synthetic datasets The first two entries in Table 4.4 summarise the performance on the two *synthetic* datasets. The *Pairs* and *PairsG* algorithm perform best in both cases, with little difference in performance between them. This is because these datasets suit the *Pairs* algorithm since there are only pairwise interactions. The *Lachiche* algorithm results in no improvement in most cases for the first dataset, and none at all for the second. The *Naive* algorithm generally results in an improvement in performance, but is outperformed by the *Greedy* and *Pairs* algorithms. This is because of the limitation of the *Naive* approach, which is prone to local minima. The *Pairs* algorithm outperforms the *Greedy* approach here because all interactions are accounted for. Referring to Table 4.5, it can be seen that the *PairsG* algorithm outperforms the *Greedy*

algorithm more often as the number of classes increases. This shows that the pairwise approach scales with increasing C, whereas the *Greedy* approach becomes more susceptible to local minima.

Real datasets Referring to Table 4.4, in general, the *PairsG* algorithm performs best overall. In the *Digits* and *Satellite* cases, the *Pairs* algorithm performs well (better than the *Greedy* approach), showing that in some cases the raw algorithm is robust. This suggests that these datasets may be dominated by pairwise interactions. In the case of the *Letter* and *Cbands* datasets, the *Pairs* approach is inferior to the *Greedy* algorithm, suggesting important higher-order interactions. The inclusion of the post-processing (*PairsG*) results in superior performance, showing that the initial weighting from the pairwise algorithm is in fact useful. The reason is that the initial *Pairs* algorithm optimises the most important interactions for the given conditions, which is a good starting point. As in the case of the *synthetic* experiments, the *Naive* algorithm improves performance rather consistently, but as expected is inferior to the more sophisticated *Greedy* approach. The *Lachiche* algorithm occasionally results in a significant improvement in performance, but most scenarios result in none at all. The *Cbands* dataset shows a scenario where the *Greedy* algorithm competes in many cases with the *PairsG* algorithm. This dataset may have many higher order interactions that are not dealt with effectively due to the pairwise limitation.

Considering Table 4.5, it can be seen that the *PairsG* algorithm is generally better than the *Greedy* approach. The *Letter* dataset results in the best *PairsG* performance, with 49 out of 50 scenarios favouring this algorithm, whereas only 70% of experiments show superiority in the *CBands* case. Perhaps the *PairsG* algorithm would perform better if more interactions were considered here (i.e. increasing the AUC threshold t_p), but this would significantly increase the computational burden.

4.2.7 Conclusion

In this paper a practical framework for generalised multiclass ROC analysis was presented, providing an extension of techniques and analyses commonly used in 2-class ROC analysis. The computational complexity of multiclass ROC analysis was discussed as a function of the number of classes and ROC resolution, showing a computational increase to the power of the number of classes (-1). This severely limits it's practical use to problems with a small number of classes. This limitation was used as the argument for the development of approximate techniques. For cost-sensitive applications, two simple algorithms were proposed that use a search paradigm, in which a new cost and prior is used to direct a search, resulting in new "optimal" operating weights. The first uses a simple approach that optimises operating weights independently, ignoring interactions, and the second uses a greedy search with random initialisations, attempting to account for interactions. Both algorithms extend linearly with the number of classes C, and are thus tractable for even high C.

Dataset	Algorithm	Min	Median	Best	Mean
PRTools8	Naive	0.000	0.027	0.108	0.030
	Greedy	0.021	0.054	0.147	0.058
	Lachiche	0.000	0.000	0.071	0.002
	Pairs	**0.023**	0.057	**0.148**	0.060
	PairsG	**0.023**	**0.058**	**0.148**	**0.061**
PRTools16	Naive	0.000	0.022	0.094	0.022
	Greedy	0.017	0.047	0.132	0.055
	Lachiche	0.000	0.000	0.000	0.000
	Pairs	**0.019**	0.050	**0.135**	**0.057**
	PairsG	**0.019**	**0.051**	**0.135**	**0.057**
Letter	Naive	0.003	0.015	0.026	0.015
	Greedy	0.018	0.027	0.039	0.028
	Lachiche	0.000	0.000	0.000	0.000
	Pairs	0.014	0.024	0.036	0.024
	PairsG	**0.021**	**0.031**	**0.042**	**0.032**
Cbands	Naive	0.000	0.017	0.045	0.019
	Greedy	0.013	0.030	0.060	0.031
	Lachiche	0.000	0.000	0.000	0.000
	Pairs	0.009	0.025	0.054	0.026
	PairsG	**0.014**	**0.032**	**0.061**	**0.032**
Digits	Naive	0.005	0.056	0.184	0.064
	Greedy	0.033	0.081	0.199	0.090
	Lachiche	0.000	0.000	0.164	0.011
	Pairs	0.034	0.082	0.199	0.094
	PairsG	**0.037**	**0.086**	**0.201**	**0.098**
Satellite	Naive	0.000	0.009	0.084	0.007
	Greedy	0.000	0.025	0.120	0.035
	Lachiche	0.000	0.000	0.082	0.005
	Pairs	0.000	0.033	**0.135**	0.040
	PairsG	0.000	**0.034**	0.134	**0.041**

Table 4.4: Comparing the various algorithms for each dataset, summarising results over 50 different cost-sensitive scenarios. Each value represents the difference between the algorithm MSU performance and the default classifier performance, showing the minimum (**Min**), median (**Median**), maximum (**Best**), and mean (**Mean**) performance across the 50 runs. Larger differences are favourable. The best performers per statistic are highlighted in bold.

Dataset	PairsG > Greedy
PRTools8	80%
PRTools16	96%
Letter	98%
Cbands	70%
Digits	88%
Satellite	74%

Table 4.5: Comparing the Pairwise algorithm with post-processing, with the Greedy algorithm, indicating the percentage of experiments in which the *PairsG* algorithm was superior.

However, these approaches do not result in a multiclass ROC, losing the various benefits, and are susceptible to local minima. The paper then presented an approximation of the multiclass ROC, called the *Pairwise Multiclass ROC*, which is tractable for high C problems. As the name suggests, this algorithm investigates interactions between operating weight pairs (2-class ROC's). In the cost-sensitive case, each ROC pair is optimised to the new priors and costs, followed by construction of the final operating weight vector. However several possible pair combinations exist, so the pairwise algorithm considers the various feasible pairings, and chooses the best one, based on minimum loss. Conceptually, this results in considering the most interacting pairs, and the most costly errors, which impact the new situation most significantly. A variety of experiments compared the various approaches, showing consistent benefits (except the *Lachiche* algorithm) of the various algorithms over a default unoptimised classifier in the cost-sensitive case. The naive approach is usually inferior to the greedy approach, and the pairwise algorithm outperformed the other algorithms in most cases, indicating that it is well formulated, and generally works well.

It is anticipated that this study will encourage new algorithms and approaches to tackling the area of multiclass ROC analysis, which will further diversify the field of statistical pattern recognition into new applications.

Bibliography

[1] N.M. Adams and D.J. Hand. Comparing classifiers when misallocation costs are uncertain. *Pattern Recognition*, 32(7):1139–1147, 1999.

[2] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press Inc., New York, first edition, 1995.

[3] A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

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

[5] R.P.W. Duin. *PRTools, A Matlab Toolbox for Pattern Recognition*. Pattern Recognition Group, TUDelft, January 2000.

[6] D.C. Edwards, C.E. Metz, and M.A. Kupinski. Ideal observers and optimal ROC hypersurfaces in N-class classification. *IEEE Transactions on Medical Imaging*, 23(7):891–895, July 2004.

[7] ELENA. European ESPRIT 5516 project. *phoneme dataset*, 2004.

[8] R.M. Everson and J.E. Fieldsend. Multi-class ROC analysis from a multi-objective optimisation perspective. *Pattern Recognition Letters, Special issue on ROC analysis*, 27, 2005.

[9] T Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters, Special issue on ROC analysis*, 27:861–874, 2005.

[10] C. Ferri, J. Hernandez-Orallo, and M.A. Salido. Volume under the roc surface for multi-class problems. *Proc. of 14th European Conference on Machine Learning*, pages 108–120, 2003.

[11] D.J. Hand and R.J. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45:171–186, 2001.

[12] J. Houtepen. Chromosome banding profiles: how many samples are necessary for a neural network classifier? *Masters Thesis, Delft University of Technology*, pages 1–120, 1994.

[13] N. Lachiche and P. Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. *Proc. 20th Int. Conf. on Machine Learning, Washington DC*, pages 416–423, 2003.

[14] T.C.W. Landgrebe and R.P.W Duin. On Neyman-Pearson optimisation for multiclass classifiers. *Sixteenth Annual Symposium of the Pattern Recognition Assoc. of South Africa*, November 2005.

[15] T.C.W. Landgrebe and R.P.W Duin. Combining accuracy and prior sensitivity for classifier design under prior uncertainty. *Structural and Syntactic Pattern Recognition, Proc. SSPR2006 (Hong Kong, China), Lecture notes in computer science vol. 4109, Springer Verlag, Berlin*, pages 512–521, August 2006.

[16] M. Li and I.K. Sethi. Confidence-based classifier design. *Pattern Recognition*, 39:1230–1240, 2006.

[17] R.A. McDonald. The mean subjective utility score, a novel metric for cost-sensitive classifier evaluation. *Pattern Recognition Letters*, 27(13):1472–1477, October 2006.

[18] C. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, 3(4), 1978.

[19] D. Mossman. Three-way roc's. *Medical Decision Making*, 19:78–89, 1999.

[20] P.M. Murphy and D.W. Aha. UCI repository of machine learning databases, ftp://ftp.ics.uci.edu/pub/machine-learning-databases. *University of California, Department of Information and Computer Science*, 1992.

[21] T. V. Pham, M. Worring, and A. W. M. Smeulders. Face detection by aggregated bayesian network classifiers. *Pattern Recognition Letters*, 23(4):451–461, February 2002.

[22] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42:203–231, 2001.

[23] A. Srinivasan. Note on the location of optimal classifiers in N-dimensional ROC space. *Oxford University Computing Laboratory Technical report PRG-TR-2-99*, October 1999.

4.3 Efficient multiclass ROC approximation by decomposition via confusion matrix perturbation analysis

This section has been accepted as 'Efficient multiclass ROC approximation by decomposition via confusion matrix perturbation analysis', by T.C.W. Landgrebe, and R.P.W. Duin, in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, February 2007.

Abstract

ROC analysis has become a standard tool in the design and evaluation of 2-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 non-ideal. Extending this to the multi-class case is attractive, conferring the benefits of ROC analysis to a multitude of new problems. Even though ROC analysis does extend 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 analyses interactions between various ROC dimensions, identifying independent classes, and groups of interacting classes, allowing the ROC to be decomposed. The resultant decomposed ROC hypersurface can be interrogated in a similar fashion to the ideal case, allowing for approaches such as cost-sensitive and Neyman-Pearson optimisation, as well as the volume under the ROC. An extensive bouquet of examples and experiments demonstrates the potential of this methodology.

4.3.1 Introduction

Receiver Operator Characteristics (ROC's) [22], [11] have become a standard tool for the design, optimisation, and evaluation of 2-class classifiers. In cost-sensitive problems the ROC can be used directly to select the best operating point based on given priors and costs [28]. Similarly Neyman-Pearson type optimisation can be carried out simply by selecting the operating point corresponding to a specified error [6]. In imprecise environments ROC analysis is particularly useful since it provides the means to compare competing models over a range of operating conditions [28]. The Area Under the ROC (AUC) [3] 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 analysed, since they are constrained to vary along the ROC [16].

However the ROC has only been studied primarily in the 2-class case. Extension to the multiclass case is attractive since it would confer the benefits of ROC analysis to many more problems in pattern recognition. Recently, a number of studies in this area have been performed. The 3-class case has been studied in [23] and [5]. In [18] we generalised the multiclass ROC using a framework involving weighting of classifier outputs, which are analogous to the 2-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 [29] the ROC convex hull method for comparing classifiers in [27] was shown to extend theoretically to the $C-$class case. The Volume under the ROC hyper-Surface (VUS), which is a generalisation 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 [8], 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 [17] we presented a simplified VUS measure that does extend with C, but these approaches are all limited to low C. The work in [26] and [13] 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- optimisation is also related to ROC analysis. In the cost-sensitive case, the classifier weight/threshold optimisation problem has been posed in an optimisation framework. In [14] and [25], greedy search approaches were developed to optimise a classifier to given costs/priors, but are prone to local minima. In [18], a naive and greedy approach was presented, as well as a data-driven approach involving the construction of 2-class ROC's between all class pairs. Classifier weights are assembled from the various ROC pairs based on the given class priors and costs. An evolutionary approach was proposed in [10], attempting to find a global solution to the optimisation problem. Other related approaches have been proposed in [4] and [2]. In [7] the theoretical extension of Neyman-Pearson optimisation to multiclass optimisation was discussed, allowing for the specification of any element in the confusion matrix. In [15] we presented an algorithm allowing 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 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, 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 optimised independently. These groups are often much smaller than the original problem, reducing computational requirements drastically. An algorithm is presented that 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.

The paper is constructed as follows: In Section 4.3.2 a multiclass analysis framework is formalised, as well as the construction of the multiclass ROC. Next Section 4.3.3 discusses the potential and consequences of ROC decomposition, showing how 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 4.3.4 via a controlled experiment, showing results on cost-sensitive experiments and VUS estimations as a function of class overlap. Section 4.3.5 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 Sec 4.3.6 that investigate cost-sensitive optimisations using the decomposition in a number of synthetic and real scenarios. Finally conclusions are presented in Section 4.3.7.

4.3.2 Notation and multiclass ROC analysis

Multiclass analysis framework

The output $p(\mathbf{x})$ of a multiclass classifier consists of C values, corresponding to classes $\omega_1, \omega_2, \ldots \omega_C$, 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 per Equation 4.20.

$$arg \max_{i=1}^{C} p(\omega_i|\mathbf{x}) \tag{4.20}$$

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), defined in Table 4.6. 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 i.e. $\xi_{i,i} = 1 - \sum_{j=1}^{C} \xi_{i,j}, i \neq j$. 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 [19] etc. In this case a confusion matrix CM is generated via application of a representative independent test set. These CM outputs are normalised by the absolute number of objects N_i per class ω_i, $N = [N_1, N_2, \ldots N_C]^T$, 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:

$$\xi_{i,j} = p(\omega_i) \int p(\mathbf{x}|\omega_i) I_j(\mathbf{x}) dx \tag{4.21}$$

Table 4.6: The multi-class 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	$\xi_{2,j}$	$\xi_{2,1}$	$\xi_{2,2}$	$\cdots$	$\xi_{2,C}$
	$\vdots$	$\vdots$		$\ddots$	
	$\xi_{C,j}$	$\xi_{C,1}$	$\xi_{C,2}$	$\cdots$	$\xi_{C,C}$

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

$$I_j(\mathbf{x}) = \begin{cases} 1 \text{ if } p(\omega_j|\mathbf{x}) > p(\omega_k|\mathbf{x}) \ \forall k, k \neq j \\ 0 \text{ otherwise} \end{cases} \tag{4.22}$$

Equation 3.2 allows any confusion matrix output to be computed, generalised for both diagonal and off-diagonal elements.

Multiclass ROC

The confusion matrix only defines the performance at a single *operating point*, valid for a single prior probability situation, and single position of the classifier *thresholds/weights*. The classifier thresholds/weights can be manipulated by weighting the classifier output $p(\mathbf{x})$ by $\Phi = [\phi_1, \phi_2, \ldots \phi_C], \phi_i \geq 0 \ \forall i$. Equation 4.20 can then be generalised as Equation 4.5. Since the class assignment decisions are relative, this implies that there are $C-1$ degrees of freedom, and one weight can be held constant[4].

$$arg \max_{i=1}^{C} \phi_i p(\omega_i|\mathbf{x}) \tag{4.23}$$

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 Equation 3.2, resulting in Equation 4.24.

$$\xi_{i,j}(\Phi) = \phi_i p(\omega_i) \int p(\mathbf{x}|\omega_i) I_j(\mathbf{x}|\Phi) dx \tag{4.24}$$

The indicator function $I_j(\mathbf{x}|\Phi)$ is as in 4.22, except each posterior is multiplied by the corresponding class weight, i.e.:

$$I_j(\mathbf{x}|\Phi) = \begin{cases} 1 \text{ if } \phi_j p(\omega_j|\mathbf{x}) > \phi_k p(\omega_k|\mathbf{x}) \ \forall k, \\ \ k = 1, 2, \ldots C, \ k \neq j \\ 0 \text{ otherwise} \end{cases} \tag{4.25}$$

[4]It is important to note that there are several manifestations of multiclass classifiers, such as one-vs-all classifiers, error-correcting codes, etc, which each have architectural parameters that can be tuned. In this paper we consider only the final classifier, i.e. 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

In the 2-class case, the ROC is monotonically increasing, so efficient generation of thresholds is typically achieved using ordering of data samples [11]. This is not the general case, so the approach taken here is to generate a combinatorial $(C-1)$ dimensional grid of weightings/thresholds, considering all possible combinations of inter-class weightings. A total of r different arbitrary weightings are used, and thus the Φ matrix is $r^{C-1} \times C$ in size, clearly demonstrating the exponential computational complexity of generalised ROC analysis ($O(r^{C-1})$). The resolution must be fine enough, and the scale of each weight 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 $r = 100$. See Figure 4.6 for an example of a 3-class problem (univariate Gaussians with unit variances, and means at -0.75, 0.00, and 0.75 respectively), showing the distribution (left), and three ROC dimensions on the right, with $r = 100$, and linear resampling.

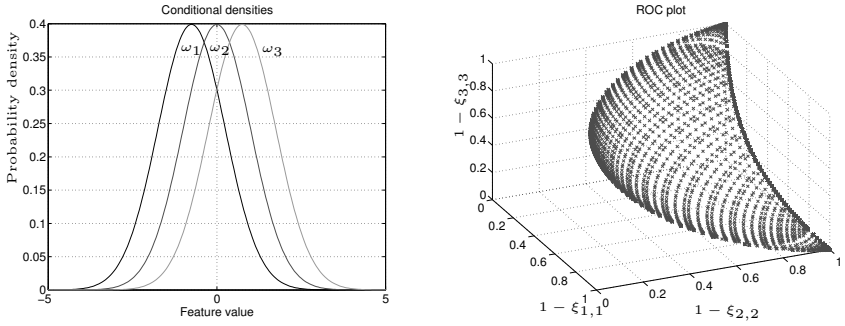


Figure 4.6: Example with 3-Gaussian classes, showing the distributions on the left, and ROC dimensions $1 - \xi_{1,1}$, $1 - \xi_{2,2}$, and $1 - \xi_{3,3}$ on the right.

4.3.3 The potential and consequences of decomposition

The exponential computational requirements (as a function of C) rule out practical construction of the multiclass ROC for high C problems. However, it may still be possible to generate an equivalent[5] 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 resultant operating characteristic is suitably accurate for the given problem.

Consider the problem in Figure 4.7, depicting a 4-class problem between ω_1, ω_2, ω_3, and ω_4, with known distributions (Gaussian distributions with unit variances, and means μ as follows: $\mu_{\omega_1} = -8$, $\mu_{\omega_2} = -5$, $\mu_{\omega_3} = 5$, and $\mu_{\omega_4} = 8$).

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

[5]'Equivalent' in this sense implies any analysis/measurements based on the new ROC would return equivalent results to the theoretical case.

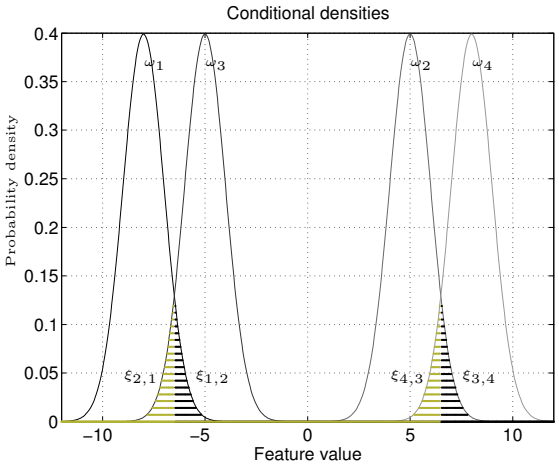


Figure 4.7: Example with known distribution, showing significant interactions.

$\xi_{i,j}$	$\xi_{i,1}$	$\xi_{i,2}$	$\xi_{i,3}$	$\xi_{i,4}$
$\xi_{1,j}$	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

(4.26)

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* e.g. 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 i.e. the impact of the variation of classifier weight $\phi_k, 1 \leq k \leq C$ on the output $\xi_{i,j}, 1 \leq i, j \leq C$. However, the example in Figure 4.7 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 resultant 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, i.e. ω_j is separable from ω_k.

For classes that are completely separable, e.g. ω_j, it holds that:

$$\Delta\xi_{i,j}(\Delta\phi_k) = 0\ \forall i, k \tag{4.27}$$

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[6], thus reducing the computational requirements from $O(r^{C-1})$ to $O(r^{C-2})$.

[6]The classifier weight ϕ_j can simply be set to an arbitrary non-zero value, e.g. 1, for all operating points.

Another type of conceivable decomposition situation is illustrated in Figure 4.7, in which groups of classes interact independently from other classes/groups e.g. ω_k and ω_l are inter-dependent, but independent of ω_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 ϕ_m $\forall m, m \neq k, l$ will have no impact on outputs $\xi_{i,k}$ $\forall i$ and $\xi_{i,l}$ $\forall i$, i.e. 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 it's constituent groups. So in this example, the original 4-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 Figure 4.8 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 $\xi_{4,3}$.

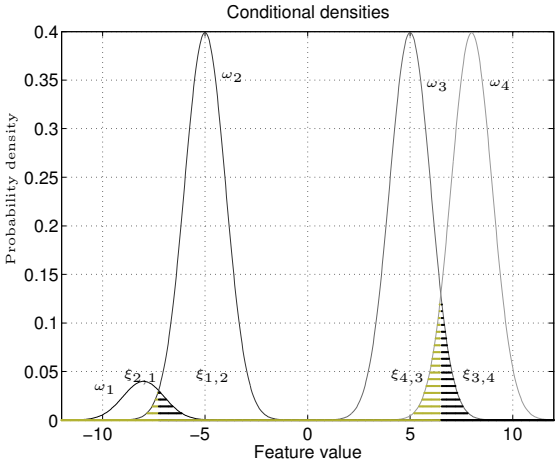


Figure 4.8: Example with known distribution, with a perturbation of ϕ_1 from 1.0 to 0.1.

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.

Case study on a 10-class problem

Consider for example the the *Digits-Zernike* dataset, consisting of 2000 examples of ten handwritten digits (from '0' to '9'), originating from Dutch utility maps (available from [24]). In this dataset, Zernike-moments have been ex-

tracted 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, using a principle component mapping (20 components retained), followed by a Bayes quadratic classifier, which is then evaluated on the remainder of the data. On the left of Figure 4.9 the resultant normalised confusion matrix is graphically shown.

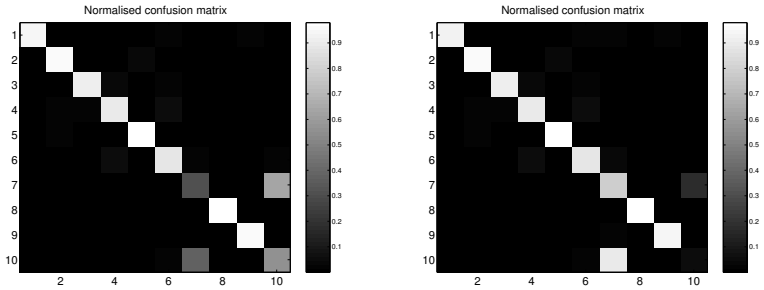


Figure 4.9: Normalised confusion matrix for the Digits dataset example, with unit weighting on the left, and a perturbation of ϕ_7 on the right.

It can be seen that most classes appear approximately separable except for the 7th and 10th, corresponding 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 normalised confusion matrix shown in the right of Figure 4.9. 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.

4.3.4 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 sub-optimal, but nevertheless useful, solution. In the cost-sensitive optimisation case, the weights obtained from the decomposed ROC could be treated as a good starting point for a post-processing search approach (see e.g. evolutionary search in [10], naive and greedy search in [18], and [25]). This could be performed to account for the smaller interactions that were ignored.

In this sub-optimal 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 Figure 4.10, illustrating a 3-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 $O(r^2)$ to an $O(r+1)$ calculation.

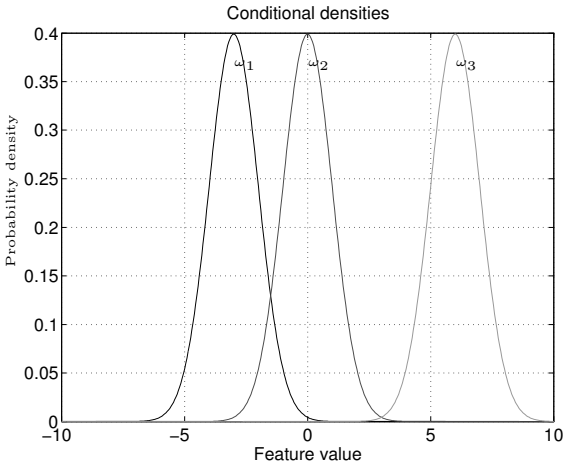


Figure 4.10: A 3-class problem, with a low degree of interaction between ω_2 and ω_3.

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: firstly a cost-sensitive optimisation scenario; and secondly by inspecting the impact on the Volume under the ROC measure.

Cost-sensitive optimisation 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 (i.e. performance is compared for 50 different operating points). The $C \times C$ dimensional cost matrix S consists of profits on the diagonal, and 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 Figure 4.11 the difference between the loss (Equation 4.28) 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 $\frac{L_d - L_{gt}}{L_{gt}}$, as well as the upper and lower quartiles (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 i.e. 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.

$$L = \sum_{i=1}^{C} P(\omega_i)(\sum_{j=1, i \neq j}^{C} \xi_{i,j} s_{ij}) - \sum_{i=1}^{C} P(\omega_i) \xi_{i,i} s_{ii} \tag{4.28}$$

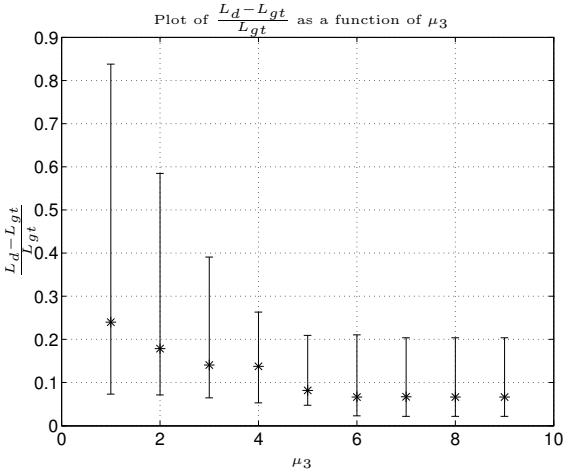


Figure 4.11: Comparing the cost-sensitive optimisation performance between the ideal and decomposed ROC techniques as the degree of interaction between ω_3 and other classes is varied. The median, and upper and lower quartiles are shown for each μ_3 value.

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 Volume Under the ROC hyper-Surface (VUS), which extends the popular 2-class Area Under the ROC measure [3] to the multiclass case. The VUS measure allows for an evaluation that encompasses all operating points. The simplified VUS measure proposed in [17] is used for this study, that considers only ROC dimensions corresponding to diagonal elements of the ROC. These performances are equivalent to the comple-

ment of a summation of off-diagonal errors in the corresponding confusion matrix row i.e. $\xi_{i,i} = 1 - \sum_{j=1}^{C} \xi_{i,j}$. The upper bound is 1.0, and the lower bound was conjectured to be $\frac{1}{C!}$. The VUS measure can be formalised as $VUS = \int \ldots \int \int \xi_{C,C} d\xi_{C-1,C-1} d\xi_{C-2,C-2} \ldots d\xi_{1,1}$. 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 [8]. 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[7]. In the 3-class example with one separable class, the VUS can be approximated as the VUS between the $[\omega_1, \omega_2]$ group (AUC in 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 Figure 4.12. In these experiments the full 3-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 2-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 [17], using 80-steps, and linear resampling. This is compared to the decomposition VUS approximation (VUS_d), in which the 2-class VUS is multiplied by 1.0 to account for the separable dimension. Figure 4.12 shows $\frac{VUS_d - VUS_{gt}}{VUS_{gt}}$ as a function of the separability of ω_3, indicating the accuracy of the VUS estimation. Note that this VUS is a performance measure i.e. 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.

4.3.5 Confusion matrix perturbation analysis

The previous sections showed that perturbing weights and analysing the subsequent confusion matrix dynamics is useful for identifying independent ROC dimensions, as well as 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 to 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 resultant confusion rate matrices Ξ as each classifier weight is independently perturbed. As a starting point, the algorithm considers a "default" un-optimised classifier (often trained using population or balanced priors) i.e. a single operating point. It is important that the default classifier is not at an extreme operating point i.e. where one or more classes has no discriminability, by ensuring $\phi_i > 0 \; \forall i$.

[7] Generalisation and proof of this is the topic of further research.

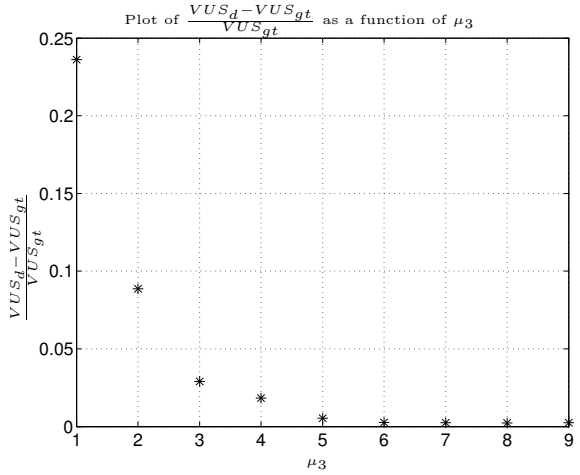


Figure 4.12: Comparing the VUS performance as in Figure 4.11, with $\frac{VUS_d - VUS_{gt}}{VUS_{gt}}$ representing the difference between the ideal and the decomposed calculation.

Algorithm

In 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 $c_j, 1 \leq j \leq c, j \neq i, c_j \neq 0$ (e.g. $c_j = 1 \ \forall j$). 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 Equation 4.29, with dimensionality $C \times C \times r$. The weight perturbation procedure is repeated for each weight, resulting in C of these Λ structures.

$$\Lambda_i = \xi_{j,k}(\Phi_i(\Theta)) \ \forall j, k \tag{4.29}$$

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

$$\Phi_i(\Theta) = \begin{bmatrix} c_1 & c_2 & \dots & c_{i-1} & \theta_1 & c_{i+1} & \dots & c_C \\ c_1 & c_2 & \dots & c_{i-1} & \theta_2 & c_{i+1} & \dots & c_C \\ \vdots & & & & & & & \\ c_1 & c_2 & \dots & c_{i-1} & \theta_r & c_{i+1} & \dots & c_C \end{bmatrix} \tag{4.30}$$

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:

$$\Theta = [10^{\alpha_1}, 10^{(\alpha_1 + \frac{\alpha_2 - \alpha_1}{r-1})}, 10^{(\alpha_1 + 2\frac{\alpha_2 - \alpha_1}{r-1})}, \dots 10^{(\alpha_1 + (r-2)\frac{\alpha_2 - \alpha_1}{r-1})}, 10^{\alpha_2}] \tag{4.31}$$

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 shown in Equation 4.32.

$$\begin{aligned} \Lambda^s_{i(j,k)} &= \Delta\Lambda_{i(j,k)} \\ &= \max(\Lambda_{i(j,k,l)}) - \min(\Lambda_{i(j,k,l)}) \forall l, 1 \leq l \leq r \end{aligned} \tag{4.32}$$

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

$$\Lambda_i^s = \begin{bmatrix} \Delta\Lambda_{i(1,1)} & \Delta\Lambda_{i(1,2)} & \dots & \Delta\Lambda_{i(1,C)} \\ \Delta\Lambda_{i(2,1)} & \Delta\Lambda_{i(2,2)} & \dots & \Delta\Lambda_{i(2,C)} \\ \vdots & & & \\ \Delta\Lambda_{i(C,1)} & \Delta\Lambda_{i(C,2)} & \dots & \Delta\Lambda_{i(C,C)} \end{bmatrix} \tag{4.33}$$

This representation now summarises 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 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 in Equation 4.34, calculating of the largest interaction per class.

$$\Lambda_i^v = [\max \Lambda^s_{i(k,1)},\ \max \Lambda^s_{i(k,2)},\ \dots \max \Lambda^s_{i(k,C)}] \forall k \tag{4.34}$$

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 binarised, with all "sensitive" entries set to 1, and 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 binarised sensitivity vectors, denoted $\Lambda_i^{v_b}$ for ω_i can be written as in Equation 4.35 for each element k of Λ_i^v, denoted $\Lambda^v_{i(k)}, 1 \leq k \leq C$.

$$\Lambda^{v_b}_{i(k)} = \begin{cases} 1\, if\, \Lambda^v_{i(k)} > t_s \\ 0\, otherwise \end{cases} \tag{4.35}$$

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 4.3.4. 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 non-zero weighting could be used for all operating points.

Illustration of algorithm

The algorithm is presented via a running example for clarity, consisting of 4 Gaussian-distributed classes $\omega_1, \omega_2, \omega_3, \omega_4$, with means occurring at $\mu_1 = -8, \mu_2 = 0, \mu_3 = -6$, and $\mu_4 = 5$, and 0.5 variances (plotted on the left of Figure 4.13).

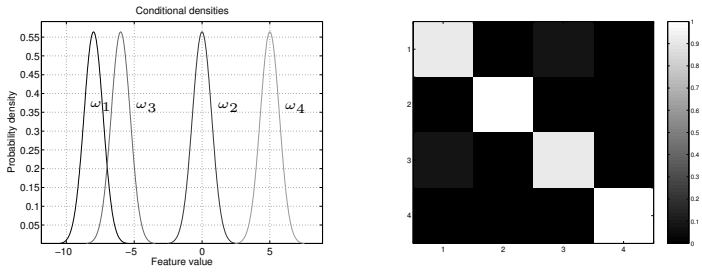


Figure 4.13: Probability density functions for the 4-class example with known distributions on the left, with default Ξ for the example on the right ($\Phi = [1\,1\,1\,1]$).

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 normalised confusion matrix Ξ for an equal prior is shown on the right of Figure 4.13, illustrating interclass errors and intraclass performances for a fixed operating point (this is referred to as the *default* confusion matrix).

Figure 4.14 presents some Λ slices for the example, shown for 3 different weightings (the 4 columns correspond to $\Lambda_1, \Lambda_2, \Lambda_3$, 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 to assess the dependence between a particular weight, and the various ROC dimensions.

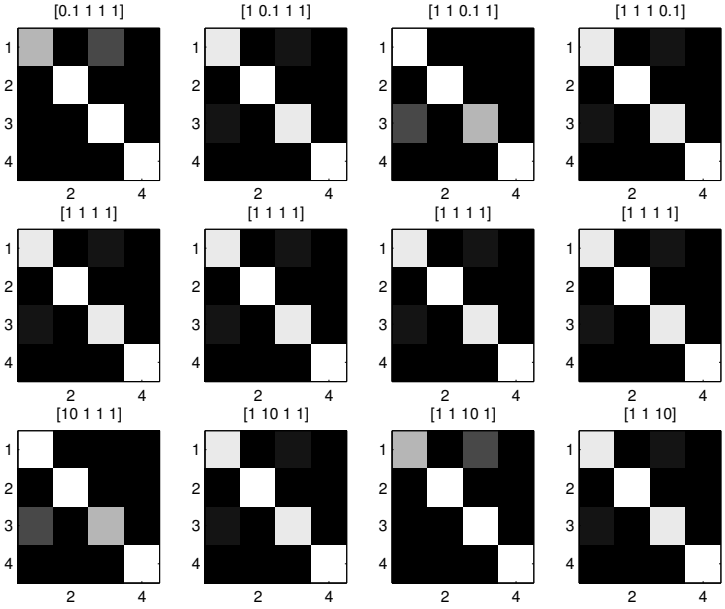


Figure 4.14: Slices of Λ for the 4-class example for a number of different Φ weightings, shown above each plot.

Next the Λ matrices are simplified into $C \times C$-dimensional sensitivity matrices Λ^s, as depicted in Figure 4.15. 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 via Equation 4.34, resulting in the following Λ^v vectors:

$$\begin{array}{rcl} \Lambda_1^v & = & [0.8554 \quad 0.0000 \quad 0.8554 \quad 0.0000] \\ \Lambda_2^v & = & [0.0000 \quad 0.0058 \quad 0.0003 \quad 0.0055] \\ \Lambda_3^v & = & [0.8554 \quad 0.0003 \quad 0.8558 \quad 0.0000] \\ \Lambda_4^v & = & [0.0000 \quad 0.0055 \quad 0.0000 \quad 0.0055] \end{array} \tag{4.36}$$

These are then binarised via Equation 4.35, resulting in Λ^{v_b}, using $t_s = 0.01$, resulting in:

$$\begin{array}{rcl} \Lambda_1^{v_b} & = & [1 \quad 0 \quad 1 \quad 0] \\ \Lambda_2^{v_b} & = & [0 \quad 0 \quad 0 \quad 0] \\ \Lambda_3^{v_b} & = & [1 \quad 0 \quad 1 \quad 0] \\ \Lambda_4^{v_b} & = & [0 \quad 0 \quad 0 \quad 0] \end{array} \tag{4.37}$$

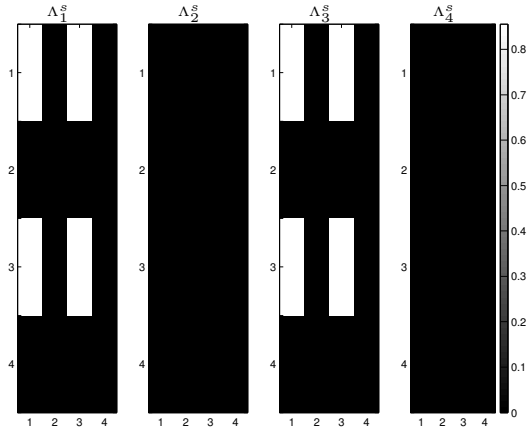


Figure 4.15: Sensitivity matrices Λ^s for the 4-class example, resulting from a sensitivity analysis of the Λ structures.

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

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 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 emphasise this rather subtle point, consider the 3-class problem in Figure 4.16, between ω_1, ω_2, and ω_3. The first class consists of a bimodal Gaussian distribution with means at 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 on the left, and a more imbalanced setting on the right.

The following Ξ results correspond to the operating points depicted in Figure 4.16:

$\xi_{i,j}$	$\xi_{i,1}$	$\xi_{i,2}$	$\xi_{i,3}$
$\xi_{1,j}$	0.9273	0.0227	0.0500
$\xi_{2,j}$	0.0142	0.9858	0.0000
$\xi_{3,j}$	0.0000	0.0000	1.0000

(4.38)

$\xi_{i,j}$	$\xi_{i,1}$	$\xi_{i,2}$	$\xi_{i,3}$
$\xi_{1,j}$	0.9508	0.0018	0.0474
$\xi_{2,j}$	0.1016	0.8984	0.0000
$\xi_{3,j}$	0.0119	0.0000	0.9881

(4.39)

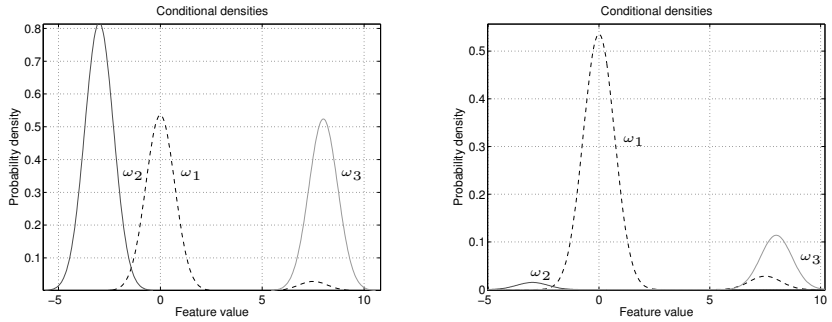


Figure 4.16: Comparing two operating points for a 3-class problem, with close to a balanced weighting on the left, and a more imbalanced weighting on the right.

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 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.

4.3.6 Experiments

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 consisting of synthetic examples that demonstrate performance in ideal circumstances, and the second consisting of realistic examples, that are not necessarily ideal.

The experimental protocol is as follows: each dataset is analysed separately, and a competitive classifier is chosen based on minimisation 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% 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 Equation 4.35) 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 (Equation 4.28) 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

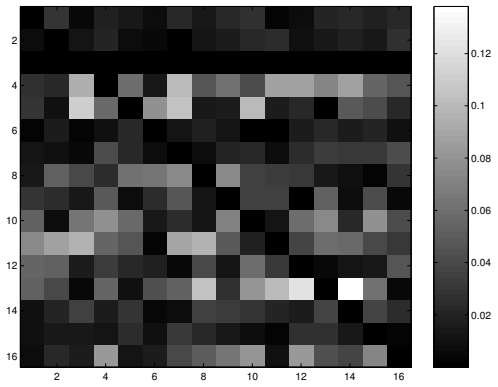
to compute the optimal classifier weightings (as was done in the 3-class case in Section 4.3.4).

The evaluation approach taken is to compare the loss obtained via the *Decomp* approach with 3 other (non ROC-based) cost-sensitive optimisation approaches. Optimisation 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 optimised by considering the overall cost per class, multiplied by priors. So for ω_i, the corresponding weight $\phi_i = p(\omega_i) \sum_{j=1}^{C} s_{ij}$. Both intra-class 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 [18], the first of which is a slight modification of the algorithm in [14]. These algorithms attempt to optimise classifier weights according to given priors and costs using a search paradigm. The *Naive* algorithm optimises each classifier weight independent 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 optimised weights. The algorithm is repeated 3 times in experiments with random initialisations in an attempt to avoid local minima. Note that other cost-sensitive optimisation algorithms are possible, as discussed in Section 4.3.1, which is currently an active area of research. A final algorithm that is implemented involves a post-processing step applied to the ROC decomposition algorithm, attempting to overcome the independence assumption of the decomposition. This post-processing step involves a "constrained" *Greedy* algorithm, in which the output of the decomposition is used as an initialisation. 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 3-iterations of the post-processing used in experiments.

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 2-class and 3-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 2-class clusters are a distance of 3.0 apart. The means of the 3-class clusters are $\sqrt{\frac{29}{4}}$ apart between the first and second class, $\sqrt{8}$ between the first and third, and $\sqrt{\frac{37}{4}}$ between the second and third. The first experiment is an 8-class problem with one 2-class cluster, and two 3-class clusters. The second experiment is a 16-class problem, repeating the same structure as the

8-class case, staggered in feature space, as shown in Figure 4.17. Finally the third experiment consists of a 40-class problem, with five repetitions of the 8-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 unoptimised classifier results in a loss of 0.3079 in this example, with the solid line in Figure 4.17 representing the respective decision boundary. The *Decomp* algorithm attempts to minimise 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:

$$\begin{aligned}\Phi^* \quad = \quad & [1.000\, 0.323,\ 0.005,\ 1.216,\ 0.651,\ 0.377,\ 0.377,\ 5.356,\\ & 0.515,\ 0.476,\ 0.761,\ 0.019,\ 0.218,\ 0.276,\ 0.037,\ 1.124]\end{aligned} \tag{4.40}$$

The dashed line in Figure 4.17 depicts the decision boundary resulting from the *Decomp* algorithm, deviating significantly from the default case.

In Table 4.7 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 parenthesis stating the number of experiments that are significantly better/worse based on a 95% confidence i.e. 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, as well as 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 datasets. The following computation reduction due to decomposition resulted for the 8-class, 16-class, and 40-class cases respectively (denoted *Synth8C*, *Synth16C*, and *Synth40C*):

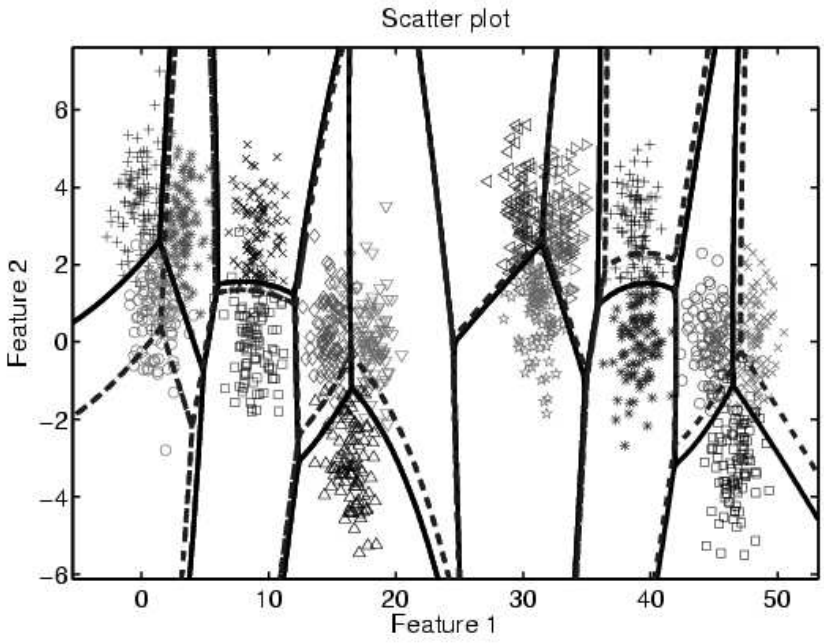


Figure 4.17: Scatter plot for the 16-class synthetic dataset. The solid line is the decision boundary of the default classifier, and the dashed line is that of the operating point provided by the *Decomp* algorithm for a particular cost.

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

It can be seen that for all 3 datasets, the *Decomp* algorithm is consistently better than the *Simple* case, 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 8-class case, showing that it is better able to account for interactions than the *Naive* algorithm. However, in the other two datasets 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 efficacy of the decomposition approach up to large numbers of classes in ideal circumstances.

Table 4.7: Results of synthetic experiments, comparing results obtained using the decomposition algorithm to the *Simple* case, as well as to the *Naive* and *Greedy* algorithms. Each comparison shows the number of experiments (out of 50) that the *Decomp* algorithm was superior (won), and inferior (lost). The number exceeding 95% statistical confidence is shown in parenthesis.

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)

Experiments with real datasets

The experiments undertaken consider a number of datasets with varying numbers of classes, as described in Table 4.8. Competing classifiers were chosen by minimisation of the equal error-rate ξ_{eq} over 10-folds. In the Table, "fish" is a Fisher projection, with the weighted pairwise Fisher mapping [20], denoted "nlfish". 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 resultant computational reduction due to decomposition for one fold (this is usually quite stable over folds, but does vary occasionally).

In many of the experiments, an ideal decomposition with a low enough computational complexity did not result, so an approximate decomposition was forced 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.

Table 4.8: Important dataset statistics, showing the dataset source, the number of objects, classes, and dimensions (d). The classifier chosen is also shown, followed by the equal error-rate ξ_{eq} (with standard deviation), the decomposition threshold used, and the resultant computational reduction due to decomposition.

Dataset	Source	Objects	C	d	Classifier	ξ_{eq}	t_s	Decomposition
Delve	[1]	2310	7	16	fish qdc	13.10(1.27)%	0.20	$2O(r^2)$
Soybean	[24]	562	15	35	ldc	5.20(2.29)%	0.05	$O(r^3)$
Dermatology	[24]	358	6	34	fish ldc	4.09(1.56)%	0.04	$O(r^2)$
Satellite	[9]	6435	6	17	fish mogc2	15.39(0.87)%	0.50	$O(r^2) + O(r)$
Segmentation	[24]	2310	7	19	fish qdc	7.92(1.15)%	0.40	$2O(r^2)$
Nist	[30]	2000	10	256	nlfish mogc2	10.90(1.37)%	0.20	$O(r^2) + 2(r)$

Comparing the *Decomp* and *Naive* results, it can again be seen that the former algorithm is usually superior, but in the *Nist* dataset 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 post-processing step is thus important

to refine results when an over-simplified decomposition is used.

Summary

The experimental results show that in some datasets, 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 datasets, 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.

Table 4.9: Results of real experiments for the *Decomp* algorithm, following the same format as Table 4.7.

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 4.10: Results of real experiments for the *DecompG* algorithm, following the same format as Table 4.7.

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)

A final result is presented in Figure 4.18 in order to give some indication of how much improvement, on average, the *DecompG* algorithm gives over the default case (i.e. no optimisation performed). A useful measure to consider here is the Mean Subjective Utility score (MSU) [21]. The MSU score scales the resultant 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 Figure 4.18, the default MSU scores are subtracted from the *DecompG* algorithm scores for each dataset. 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%, which illustrates how beneficial the optimisation can be.

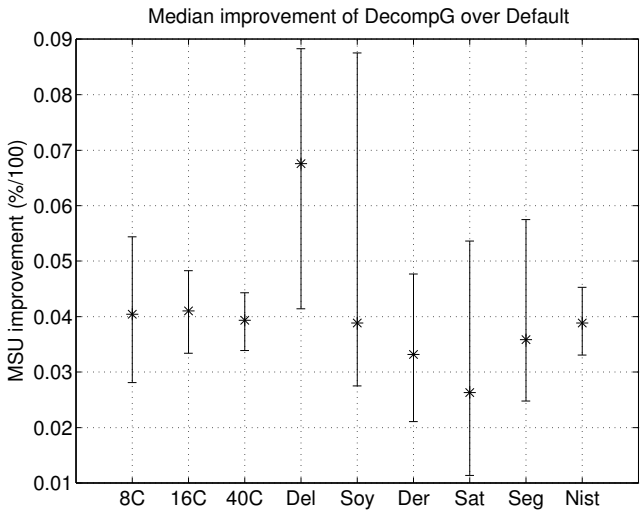


Figure 4.18: Plot showing the median improvement (%) in performance that the *DecompG* cost-sensitive algorithm gives over the default case, across the various datasets.

4.3.7 Conclusion

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 optimisation experiments, and via a simplified volume under the ROC hypersurface study. Approximate decomposition allows for a more efficient ROC construction, which is required for many real problems.

An algorithm has been proposed that 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 optimisation 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 up to 40 classes, outperforming the unoptimised classifier significantly in all experiments, and outperforming two search-based techniques. Experiments with real datasets showed that the decomposition approach is generally significantly better than the unoptimised case. In many cases it was found that the search-based approaches competed or dominated, attributed to oversimplified decomposition (required for computational reasons). A modified algorithm was also implemented that uses a search approach as a post-processing step. This proved to perform better, helping to overcome the limitation of the over-simplification.

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, generalising the various useful tools used in 2-class ROC analysis to multiclass problems.

Bibliography

[1] Delve datasets, image segmentation. *The University of Toronto, http://www.cs.toronto.edu*.

[2] N. Abe, B. Zadrozny, and J. Langford. An iterative method for multi-class cost-sensitive learning. *Proceedings of the 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 3–11, August 2004.

[3] A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[4] P. Domingos. MetaCost: A general method for making classifiers cost-sensitive. *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 155–164, 1999.

[5] S. Dreisetl, S. Ohno-Machado, and M. Binder. Comparing trichotomous tests by three-way ROC analysis. *Medical Decision Making*, 20(3):323–331, 2000.

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

[7] D.C. Edwards, C.E. Metz, and M.A. Kupinski. Ideal observers and optimal ROC hypersurfaces in N-class classification. *IEEE Transactions on Medical Imaging*, 23(7):891–895, July 2004.

[8] D.C. Edwards, C.E. Metz, and R.M. Nishikawa. The hypervolume under the ROC hypersurface of "near-guessing" and "near-perfect" observers in N-class classification tasks. *IEEE Transactions on Medical Imaging*, 24(3):293–299, March 2005.

[9] ELENA project. European ESPRIT 5516 project. *Satimage dataset*, 2004.

[10] R.M. Everson and J.E. Fieldsend. Multi-class ROC analysis from a multi-objective optimisation perspective. *Pattern Recognition Letters, Special issue on ROC analysis*, 27, 2005.

[11] T Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters, Special issue on ROC analysis*, 27:861–874, 2005.

[12] C. Ferri, J. Hernandez-Orallo, and M.A. Salido. Volume under the roc surface for multi-class problems. *Proc. of 14th European Conference on Machine Learning*, pages 108–120, 2003.

[13] D.J. Hand and R.J. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45:171–186, 2001.

[14] N. Lachiche and P. Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. *Proc. 20th Int. Conf. on Machine Learning, Washington DC*, pages 416–423, 2003.

[15] T.C.W. Landgrebe and R.P.W Duin. On Neyman-Pearson optimisation for multiclass classifiers. *Sixteenth Annual Symposium of the Pattern Recognition Assoc. of South Africa*, November 2005.

[16] T.C.W. Landgrebe and R.P.W Duin. Combining accuracy and prior sensitivity for classifier design under prior uncertainty. *Structural and Syntactic Pattern Recognition, Proc. SSPR2006 (Hong Kong, China), Lecture notes in computer science vol. 4109, Springer Verlag, Berlin*, pages 512–521, August 2006.

[17] T.C.W. Landgrebe and R.P.W. Duin. A simplified extension of the area under the ROC to the multiclass domain. *Seventeenth Annual Symposium of the Pattern Recognition Association of South Africa*, November 2006.

[18] T.C.W. Landgrebe and R.P.W. Duin. Approximating the multiclass ROC by pairwise analysis. *Pattern Recognition Letters*, 28(13):1747–1758, 2007.

[19] M. Li and I.K. Sethi. Confidence-based classifier design. *Pattern Recognition*, 39:1230–1240, 2006.

[20] M. Loog, R.P.W. Duin, and Haeb-Umbach R. Multiclass linear dimension reduction by weighted pairwise fisher criteria. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(7):762–766, 2001.

[21] R.A. McDonald. The mean subjective utility score, a novel metric for cost-sensitive classifier evaluation. *Pattern Recognition Letters*, 27(13):1472–1477, October 2006.

[22] C. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, 3(4), 1978.

[23] D. Mossman. Three-way roc's. *Medical Decision Making*, 19:78–89, 1999.

[24] P.M. Murphy and D.W. Aha. UCI repository of machine learning databases, ftp://ftp.ics.uci.edu/pub/machine-learning-databases. *University of California, Department of Information and Computer Science*, 1992.

[25] D.B. O'Brien and R.M. Gray. Improving classification performance by exploring the role of cost matrices in partitioning the estimated class probability space. *Proceedings of the ICML2005 workshop on ROC analysis in machine learning, Bonn, Germany*, 2005.

[26] F. Provost and P. Domingos. Well trained PETs: Improving probability estimation trees. *CeDER Working Paper IS-00-04, Stern School of Business, New York University NY 10012*, 2001.

[27] F. Provost and T. Fawcett. Analysis and vizualization of classifier performance: comparison under imprecise class and cost distributions. *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Miniing (KDD-97)*, pages 43–48, 2001.

[28] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42:203–231, 2001.

[29] A. Srinivasan. Note on the location of optimal classifiers in N-dimensional ROC space. *Oxford University Computing Laboratory Technical report PRG-TR-2-99*, October 1999.

[30] C.L. Wilson and M.D. M.D. Garris. Handprinted character database 3. *National Institute of Standards and Technology, Advanced Systems division*, 1992.

Chapter 5

Operating characteristics for classifier design in ill-defined problems

5.1 Overview

In some pattern recognition problems, a well-defined "target" class is to be discriminated from a "non-target" class that is less well-defined. For example, new unseen clusters may occur in the "non-target" class. One strategy to deal with this situation is to use one-class classifiers [5] that model the "target" class, and enclose the feature space occupied by this class, providing protection against a poorly defined "non-target" class. A second strategy is to use the distance-based reject-option [2], that uses a supervised classifier trained between known classes, with an internal classifier thresholding to enclose the classes of interest, providing protection from unseen classes. This thresholding could be based on, for example, class conditional density estimates, or distances in the case of classifiers such as nearest-neighbour, or support vector classifiers.

The objective in these problems is good discrimination between known classes, and effective rejection of objects from new/unseen classes. The first part of this chapter presents an alternative strategy to perform this task. A two-stage classifier is used, consisting of a one-class and a supervised classifier, which perform the tasks of rejection and classification respectively. The advantage of this approach is that models can be optimised to perform the two respective tasks separately. The second part of this chapter applies the convenience of operating characteristics to this area in order to investigate the interaction between classification and rejection performance. This has been a neglected topic to date, but is nevertheless important because increasing rejection performance is at some expense of classification performance (and vice-versa). Operating characteristics can indicate an appropriate threshold selection.

Important further work in this area involves the evaluation of the reject performance. Since the "non-target" information is not reliable, a typical approach used is to measure the volume occupied by the target class description (lower volumes are less likely to accept randomly-distributed "non-target" objects). This becomes challenging as the dimensionality increases, and thus approaches such as those described in [6] and [3] have been used to date. Another interesting open area involves inspecting the sensor domain, and it's relation to the feature-space representation. Since real-world sensors, and possibly representation schemes, are bounded, this bounds the potential location of "non-target" objects in feature space. Incorporating this information could be important for improving rejection-system design. Another related topic that has potential to tackle many new problems in a pattern recognition framework is domain-based classification [4], and the related topic of sensor-drift [1].

Bibliography

[1] K. Copsey and A. Webb. Classifier design for population and sensor drift. *Joint IAPR Workshops on Syntactical and Structural Pattern Recognition, and Statistical Pattern Recognition*, pages 744–752, August 2004.

[2] B. Dubuisson and M. Masson. A statistical decision rule with incomplete knowledge about classes. *Pattern Recognition*, 26(1):155–165, 1993.

[3] P. Juszczak. Learning to recognize, a study on one-class classification and active learning. *PhD Thesis, Delft Technical University, The Netherlands, ISBN 978-90-9020684-4*, 2006.

[4] P. Juszczak, D.M.J. Tax, S. Verzakov, and R.P.W. Duin. Domain-based LDA and QDA. *18th International Conference on Pattern Recognition (ICPR 2006), Hong Kong, China*, pages 788–791, August 2006.

[5] D.M.J. Tax. One-class classification. *PhD thesis, Delft Technical University, The Netherlands*, June 2001.

[6] D.M.J. Tax and R.P.W. Duin. Uniform object generation for optimizing one-class classifiers. *Journal for Machine Learning Research*, pages 155–173, 2001.

5.2 A combining strategy for ill-defined problems

This section has been published as 'A combining strategy for ill-defined problems', by T.C.W. Landgrebe, D.M.J. Tax, P. Paclík, and R.P.W. Duin, and C.M. Andrew, in *Fifteenth Annual Symposium of the Pattern Recognition Association of South Africa*, 57-62, November 2004

Abstract

In this paper we present a combining strategy to cope with the problem of classification in ill-defined domains. In these cases, even though a particular *target* class may be sampled in a representative manner, an *outlier* class may be poorly sampled, or new *outlier* classes may occur that have not been considered during training. This may have a considerable impact on classification performance. The objective of a classifier in this situation is to utilise all known information in discriminating, and to remain as robust as possible to changing conditions. A classification scheme is presented that deals with this problem, consisting of a sequential combination of a one-class and multi-class classifier. We show that it can outperform the traditional classifier with reject-option scheme, locally selecting/training models for the purpose of optimising the classification and rejection performance.

5.2.1 Introduction

Consider a problem in which a *target* class is to be discriminated with respect to an *outlier* class. In many applications, both classes are sampled as a set of measurements in order to construct a training set that represents the class. A classifier can then be designed, for example, by estimating the class conditional densities for both classes. Good estimates allow for an optimal tradeoff to be made between the classes. However in some applications some classes may not be well-defined. In this paper we assume that the *target* class is well represented, but the *outlier* class is not. This may be due to a variation in *outlier* class distribution, such as *sensor drift* [3], or new *outlier* classes may be present that were not represented during training. Examples of this phenomenon include:

- Diagnostic problems in which the objective of the classifier is to identify abnormal operation (*outlier* class) from normal operation (*target* class) [4]. It is often the case that a representative training set can be gathered for the *target* class, but due to the nature of the problem the *outlier* class cannot be sampled in a representative manner. For example in machine fault diagnosis [12] a destructive test for all possible abnormal states may not be feasible.

- Recognition systems often involve a detection and classification stage. An example is road sign classification, in which a classifier needs not only to discriminate between examples of road sign classes, but must also reject non-sign class examples [7]. Gathering a representative set of non-signs may not be possible. Similarly face detection [9], where a classifier must deal with well-defined face classes, and an ill-defined non-face class, as well as handwritten digit recognition [6], where non-digit examples are a serious issue.

The goal of a classifier in these cases is to obtain a high true positive rate and low false positive rate, with respect to the *target* class. Even though the *outlier*

class is poorly defined, we would still like to make use of all knowledge that exists for the problem (to account for known class overlaps). Thus the objective is to obtain a high *classification performance*, and robustness to changes in the *outlier* class (referred to as *rejection performance*). Formalisation and consequences of this problem are given in Section 5.2.2.

Previous work in this area has typically been the *classifier with reject-option*, first proposed by Chow in [2], often called the *ambiguity reject-option*. In this reject-option, when the cost of misclassification is higher than the cost of rejection, the example in question should be rejected, based on thresholding of the posterior probability. This reject-option is applicable for handling ambiguity between classes (examples close to the *target* class), which is not of interest here. In this paper we are interested in rejecting examples occurring far away from the *target* class. Dubuisson and Masson proposed the *distance reject-option* in [4]. This rejection scheme was designed to cope with the condition in which new classes are present that are not represented during training, introducing a different type of *reject* class ω_r. New examples situated a particular distance (based on a reject threshold t_d) from known class centroids are rejected. A similar procedure can be applied to density-based classifiers, except here the class conditional density is thresholded. In this way a *closed* decision surface is obtained, providing protection against new unseen classes[1]. New classes will be rejected if they fall outside the class description. Thus to minimise the probability of accepting examples from class ω_r, assuming they are uniformly distributed in feature space, the volume of the description should be minimised. The reject-option is discussed further in Section 5.2.3.

The limitation of the reject-option approach is that a model chosen for good classification performance does not necessarily imply good rejection performance. The opposite is also true. Improved performance may result from a practitioner viewpoint if an adequate evaluation methodology is used. However as will be discussed later, since the same model is used for classification and rejection, we may have to sacrifice the performance of one for the other. In this paper we present a classification strategy that can in some cases alleviate this situation, consisting of a sequential combination of one-class and multi-class classifiers (called *SOCMC*). The proposed 2-stage scheme allows both rejection and classification performance to be explicitly modified by varying the respective models and representations. Thus a classifier model can be designed to obtain good performance on known classes, and a separate classifier model to improve robustness with respect to unknown classes. The SOCMC is discussed in Section 5.2.4.

A number of experiments are performed to investigate the SOCMC approach in Section 5.2.5. All experiments benchmark SOCMC results with the distance-based reject option, as well as with traditional discriminant-based approaches. Experiments are performed on a number of real datasets, showing the applicability of the new approach. Finally, conclusions are given in Section 5.2.6.

[1] This thresholding of a single class model is equivalent to one-class classification [10].

5.2.2 Ill-defined problems

To formalise this problem, we assume that there is a well defined *target* class ω_t, and the *outlier* class is composed of two classes ω_o and ω_r, where the former consists of known class information, and the latter of unknown information (called the *reject* class). Note that in this setup, we classify examples considered to be either ω_o and ω_r as *outlier*. Examples of each class are composed of vectors of measurements $\mathbf{x}$ with dimensionality d. It is assumed that $\mathbf{x}$ is represented by a feature space χ (later we discuss classifiers that operate on the data in new feature spaces, consisting of various mappings of the original space χ). The unconditional density $p(\mathbf{x})$ can then be written as in Equation 5.1.

$$p(\mathbf{x}) = p(\omega_t)p(\mathbf{x}|\omega_t) + p(\omega_o)p(\mathbf{x}|\omega_o) + p(\omega_r)p(\mathbf{x}|\omega_r) \tag{5.1}$$

To evaluate classifiers in this situations, two performance measures are of interest:

1. The classification performance (performance between known classes/data), denoted perf(ω_t, ω_o).

2. The rejection performance (performance between the ω_t and ω_r), denoted perf(ω_t, ω_r).

Ideally both perf(ω_t, ω_o) and perf(ω_t, ω_r) should be high. Note that estimation of perf(ω_t, ω_r) is not straightforward, since this class is by definition absent during training. In the experimental Section 5.2.5 a methodology is given to provide some estimate of this. In Figure 5.1 an example of this problem is shown, demonstrating the weakness of general discrimination approaches with respect to this problem. Here a synthetic dataset has been constructed in two dimensions. In the left image, the training set consisting of ω_t and ω_o is shown, upon which a Bayes quadratic classifier is shown. In the right image the testing situation is shown, in which a new class ω_r is present. The classifier is clearly not robust to these changes in conditions.

Two classification approaches are utilised in this paper. The first are multi-class classifiers (sometimes referred to as MCC's/discriminators). In this paper, we deal specifically with two-class discriminant classifiers, denoted D_{MCC}. A classifier trained on ω_t and ω_o can be defined as in Equation 5.2, with $\hat{p}(\omega|\mathbf{x})$ representing an estimate of the posterior probability of class ω. These classifiers result in an open decision boundary, since it is assumed that ω_t and ω_o are well represented.

$$D_{MCC} : \begin{cases} \textit{target} & \text{if } \hat{p}(\omega_t|\mathbf{x}) > \hat{p}(\omega_o|\mathbf{x}) \\ \textit{outlier} & \text{otherwise} \end{cases} \tag{5.2}$$

The second classification approach used is one-class classification (sometimes referred to as OCC's), denoted D_{OCC} [10]. These classifiers are trained on only a single class, resulting in a closed description of the class density or domain. No assumptions about other classes are made, and thus these classifiers do not make a trade-off between overlapping classes. The decision boundary is however

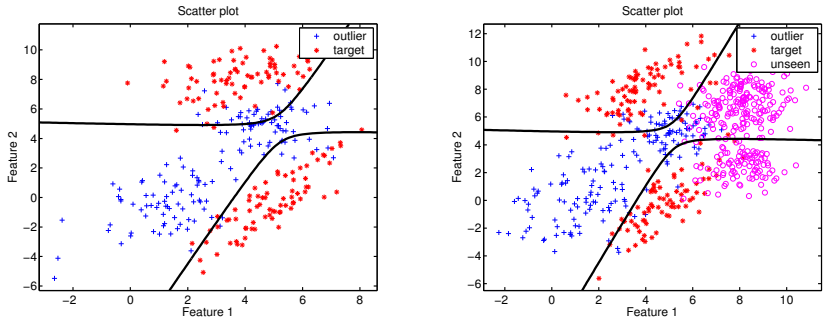


Figure 5.1: Illustrating a discrimination classifier applied to the problem in which a new unseen class is present in testing. The left plot shows the classifier decision boundary for the training data, and the right plot for the testing data, in which a new class ω_r is present. A Bayes quadratic classifier is used.

constrained/closed, i.e. all objects situated outside the class description are rejected as outliers, providing protection against new, unseen classes. The OCC description/model is trained, with some allowance made for outliers in the training set by adjusting a decision threshold θ. The D_{OCC} can be written as in Equation 5.3, classifying all objects as either *target* or *outlier*.

$$D_{OCC} : \begin{cases} target & \text{if } \hat{p}(\mathbf{x}|\omega_t) > \theta \\ outlier & \text{otherwise} \end{cases} \tag{5.3}$$

5.2.3 The classifier with reject-option

As previously mentioned, the original reject option (*ambiguity* reject) [2] rejects objects that are considered to be ambiguous, based on a threshold t_d. For an incoming test object, the classifier assigns a class label. The relevant posterior of the assigned class is examined and compared to t_d. Examples are either assigned to an $ACCEPT$ region $\Re_{accept}$ or $REJECT$ region $\Re_{reject}$, as shown in Equation 5.4.

$$\begin{aligned} \Re_{accept} &= \{\mathbf{x}| \max_i p(\omega_i|\mathbf{x}) \geq t_d\},\ i \in \{t, o\} \\ \Re_{reject} &= \{\mathbf{x}| \max_i p(\omega_i|\mathbf{x}) < t_d\},\ i \in \{t, o\} \end{aligned} \tag{5.4}$$

With the *distance reject option*, the conditional density of the class of interest is thresholded, resulting in a *closed* decision boundary[2], providing protection against unseen classes. Again a two-stage procedure is undertaken. In the first stage an example is assigned to a particular class $\omega_i, i = t, o$, referring to *target* and *outlier*, using Bayes rule. In the second step, if the example

[2]For classifiers that are not density-based such as k-Nearest Neighbour, Dubuisson and Masson proposed to reject based on the mean distance to the k nearest neighbours. In this case a meaningful threshold should be chosen based on the scale of the distances.

has been assigned to the *target* class, the conditional probability $p(\mathbf{x}|\omega_t)$ is thresholded via a reject threshold t_d. Examples exceeding this threshold are rejected. Examples are either assigned to an $ACCEPT$ or $REJECT$ region, $\Re_{accept}$ and $\Re_{reject}$ as shown in Equation 5.5.

$$\begin{aligned} \Re_{accept} &= \{\mathbf{x}|p(\mathbf{x}|\omega_i) \geq t_d\},\ i \in \{t, o\} \\ \Re_{reject} &= \{\mathbf{x}|p(\mathbf{x}|\omega_i) < t_d\},\ i \in \{t, o\} \end{aligned} \tag{5.5}$$

The *distance* reject option is illustrated on a simple example in Figure 5.2.

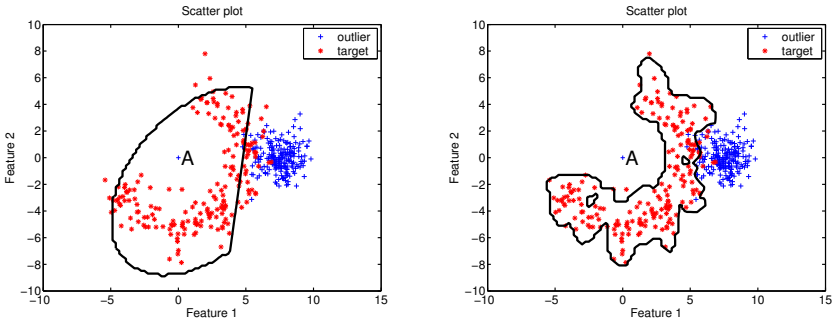


Figure 5.2: Illustrating the *distance* reject option classifier in a two-class 2D example, showing a linear classifier model in the left plot which results in good classification performance, but poor rejection performance. The right image depicts a more complex mixture-of-Gaussians classifier, resulting in good rejection, but poor classification. The decision boundary indicates the threshold used for class assignment. Poor models have purposefully been chosen for the sake of illustration to simulate realistic conditions.

The left plot shows a model based on a linear classifier, and the right image a mixture-of-Gaussians classifier with 15 mixtures. It is clear that a closed boundary results, and the trade-off between known classes is accounted for. We discuss two situations that could lead to sub-optimal performance.

In the first situation we discuss the practitioner. If the practitioner designs a classifier based on knowledge of the ω_t and ω_o classes only, a situation such as that depicted in the left figure may result. Here the classifier obtains near optimal classification performance, but since the model is not chosen explicitly to fit the *target* class distribution, sub-optimal rejection results. Thus we propose to evaluate classifiers in these situations based on both classification and rejection performance. This may lead to choosing more appropriate models. For example some classifiers focus on discrimination only, discarding domain information (e.g. support-vector classifier). A better choice would be to choose models modeling the distribution (e.g. mixture-of-Gaussians density estimation).

In the second situation we assume the practitioner is aware of an adequate evaluation methodology. In this case the practitioner will focus on obtaining

the best-possible rejection/classification performance. In real problems, typical limitations are that the training set size is limited, the input dimensionality high, and computation time limited. In these situations, choosing a model that results in high classification performance (i.e. focus on known overlapping regions) may be at the expense of a worse performance in terms of rejection performance e.g. the class conditional density may be well estimated in the overlapping region, but poor in other areas. This is depicted in the left plot of Figure 5.2 where a new *outlier* example marked A on the plot will be incorrectly classified as *target*. Similarly, the classification performance may be compromised for the case in which a model is chosen for good rejection performance (right plot). In Section 5.2.4 a classification scheme is presented in which different models can be selected/trained explicitly for classification and rejection respectively. We argue that in some cases it is better to choose a local model suitable to perform the classification, and another for rejection. This flexibility is lacking in the reject-option case.

5.2.4 Sequential combining of a one-class and multi-class classifier

We present a classification scheme here consisting of the sequential combining of one-class and multi-class classifiers (SOCMC). The rationale is that the class model and representation used in the first stage (denoted D_{OCC}) can be explicitly chosen for the purpose of rejection i.e. between ω_t and ω_r. In a similar way the second stage classifier D_{MCC} can be chosen locally in the area of known overlap to obtain good classification performance between known classes, i.e. between ω_t and ω_o. The SOCMC classifier is depicted in the block diagram in Figure 5.3. In the first stage, the one-class classifier D_{OCC} attempts

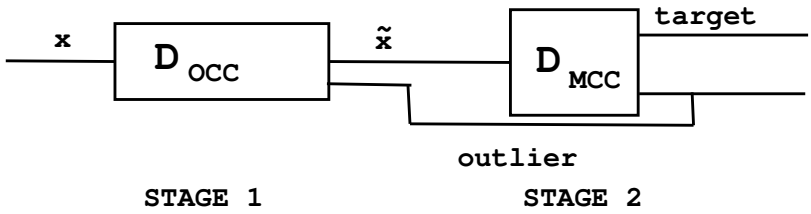


Figure 5.3: Block diagram of the *SOCMC* classifier. The first stage classifier D_{OCC} consists of an OCC, trained on the well defined *target* class. The second stage classifier D_{MCC} is a multi-class discriminant trained on examples considered to be *target* by the first stage.

to detect all *target* examples from $p(x)$, given a test set. A one-class classifier [10] is appropriate for this stage since it protects against unseen classes ω_r, and capitalises on the knowledge that the *target* class is well defined by the training

set[3]. At this stage it is not important if examples of the class ω_o are incorrectly accepted, since we rely on this discrimination in the second stage. Thus it is assumed that the output of D_{OCC}, denoted $\tilde{\mathbf{x}}$ will consist only of examples of class ω_t and ω_o, with all ω_r having been rejected (as well as ω_o examples that do not overlap with the *target* OCC description.).

Note that both the representation and class description model can be selected/trained to improve the rejection performance perf(ω_t, ω_r). The D_{OCC} represents the input data $\mathbf{x}$, derived from the feature space χ, by a new representation χ_{OCC} (D_{OCC} consists of both a representation and classification stage). The classifier can thus be written as $D_{OCC}(\mathbf{x}_{OCC})$, defined as in Equation 5.3 for class ω_t. The output $\tilde{\mathbf{x}}$ is then shown in Equation 5.6.

$$\tilde{\mathbf{x}} = \{\mathbf{x} | D_{OCC}(\mathbf{x}_{OCC}) = target\} \tag{5.6}$$

The output $\tilde{\mathbf{x}}$ is then applied to the second stage classifier D_{MCC}. Note that D_{OCC} is used to select objects for the second stage. We still have the opportunity to optimise the representation and model selection used in the second stage. Thus $\tilde{\mathbf{x}}$ is used by D_{MCC} in the original representation χ. The D_{MCC} classifier is trained on the data $\tilde{\mathbf{x}}$, which is assumed to be a mixture of data from ω_t and ω_o only, which are represented by the training set. A discriminator is thus trained, with the objective of obtaining an optimal trade-off in terms of class overlap. As with the D_{OCC}, the representation and classification model can be chosen, but in this case for the purpose of optimising the classification performance perf(ω_t, ω_o). A model is trained focused on the local region, specified by a training dataset $\tilde{(\mathbf{x})}_{tr}$. The input data $\tilde{\mathbf{x}}$ that is represented by χ is now mapped to a new representation space χ_{MCC}, resulting in the classifier $D_{MCC}(\mathbf{x}_{\tilde{MCC}})$, defined as in Equation 5.2 between classes ω_t and ω_o. The final *SOCMC* classifier, denoted D_{SOCMC} is defined in Equation 5.7.

$$D_{SOCMC}(\mathbf{x}|D_{OCC}, D_{MCC}) = \begin{cases} outlier \text{ if } D_{OCC}(\mathbf{x}) = outlier \\ D_{MCC}(\tilde{\mathbf{x}}_{MCC}) \text{ otherwise} \end{cases} \tag{5.7}$$

We illustrate the operation of the *SOCMC* classifier in the same situation as in Section 5.2.3, in Figure 5.4. We noticed that the classifier D1 in the left plot of Figure 5.2 resulted in high classification performance, and low rejection performance. The opposite was true for the classifier D2 in the right plot. In the SOCMC classifier, we select/train specific local models for the purposes of classification and rejection respectively, illustrating that the SOCMC can in some cases improve performance. In this example the model used for D1 is chosen for the D_{MCC} stage (i.e. a linear classifier), and the D2 model is used for the D_{OCC} stage (Mixture-of-Gaussians with 6 mixtures). This classifier results in a good classification and rejection performance. A number of training considerations need to be made for the *SOCMC* classifier:

[3]New classes will be rejected if they fall outside the class description. Thus to minimise the probability of accepting examples from class ω_r, assuming they are uniformly distributed, the volume of the description should be minimised.

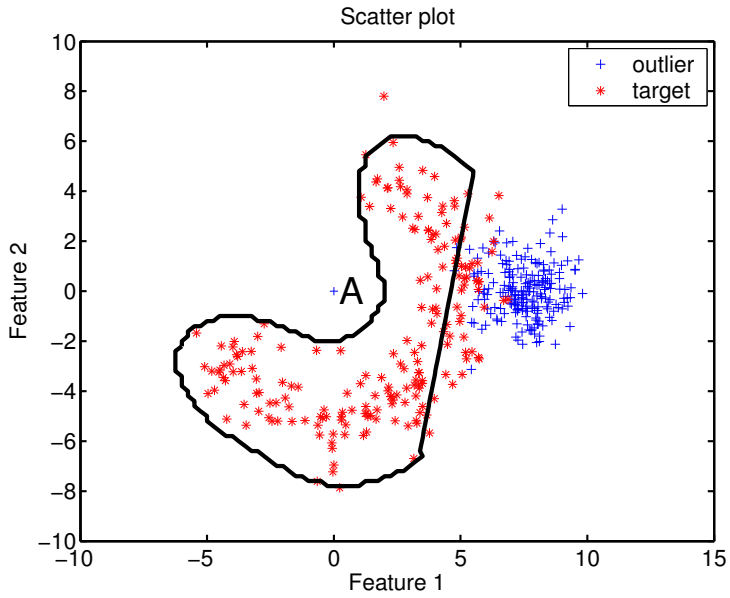


Figure 5.4: Illustrating the *SOCMC* classifier. A linear model has been chosen locally in the area of overlap for good classification performance, and a Gaussian-mixture model with 15 mixtures is used for rejection, showing that example A is correctly classified.

- Training set size for D_{MCC}: If a training set $\mathbf{x}_{tr}$ is used, only a subset $\tilde{\mathbf{x}}_{tr}$ will be available for the D_{OCC}. If $\mathbf{x}_{tr}$ is small, or $\tilde{\mathbf{x}}_{tr} \ll \mathbf{x}_{tr}$, there may not be sufficient samples to train D_{MCC}. This may limit the complexity of the model/representation used. Alternatively the entire $\mathbf{x}_{tr}$ could be used to train the D_{MCC}.
- Training technique: The *SOCMC* classifier is analogous to the trained combiner used in classifier combining, as discussed in [5]. If the same training set $\mathbf{x}_{tr}$ is used to train both D_{OCC} and D_{MCC}, the D_{SOCMC} could overfit to the noise in the training set. An alternative training strategy could be to split $\mathbf{x}_{tr}$ into two independent training sets $\mathbf{x}_{tr1}$ and $\mathbf{x}_{tr2}$, with the first used to train D_{OCC}, and the second used to train D_{MCC}. This may generalise better, but may actually be worse than the former strategy when the training set size is small.

5.2.5 Experiments

Evaluation

Since the exact nature of the *outlier* conditional distribution cannot be predicted in advance, estimating $\text{perf}(\omega_t, \omega_r)$ is not straight forward. We propose

an evaluation method to provide some confidence as to the robustness of the classifier, and to compare classifiers. The evaluation assumes a uniform *outlier* distribution. This test allows a classifier to be evaluated assuming that *outlier* examples can occur anywhere in feature space around the *target* class. It provides a measure for how well the classifier protects the *target* class (in the respective feature space) from changing conditions. However for real high dimensional problems, the number of artificial examples to be generated may be computationally prohibitive, so two methods of artificial data generation are used in real experiments to attempt to overcome this problem:

1. In the first method, called $\text{perf}_{a1}(\omega_t, \omega_r)$, a number of *outlier* examples are artificially generated uniformly in a sphere around a subspace of the *target* class [11]. Here examples are generated within a PCA (Principal Component Analysis) subspace. The original data is scaled to unit variance, and the artificial data is then generated within this space with a radius of 1.1 of the covariance of the *target* class. These can be mapped into the original space by an inverse of the PCA mapping.

2. Similar to the previous analysis, except data is generated in the original representation, following a Gaussian distribution. Here examples are generated around the *target* class, using an enlarged covariance matrix of the *target* class. The covariance matrix is enlarged by a fraction of 1.5 (this is simply a multiplication of the covariance matrix to spread the new generated examples further). The test is called $\text{perf}_{a2}(\omega_t, \omega_r)$.

The $\text{perf}(\omega_t, \omega_o)$ measure relates to the known classes ω_t and ω_o. This performance is approximated using standard techniques. For all experiments a 20-fold cross-validation procedure is carried out, and the primary performance measure used is the AUC (Area under the Receiver-Operator Curve). The variance of the estimates is depicted in terms of the standard deviation. To summarise, the following performance measures are computed for each experiment:

- $\text{perf}(\omega_t, \omega_o)$, estimated using cross-validation with 20-folds, computing the respective AUC.

- $\text{perf}_{a1}(\omega_t, \omega_r)$, estimated using 20-fold cross-validation procedure. In testing, for each fold an independent *target* portion of $\mathbf{x}$ is used, together with the generated artificial *outlier* data that was not used for training. Again the AUC is computed.

- $\text{perf}_{a2}(\omega_t, \omega_r)$, estimated as per $\text{perf}_{a1}(\omega_t, \omega_r)$.

Dataset description

A number of real-world datasets are used in the experimentation. These datasets have been selected based on their relevance to this problem. The following datasets are used:

1. *Face-amsterdam (Face)*: This dataset consists of a face class ω_t, and non-face class ω_o, and is described in [9], and downloaded at [8]. Each face is stored as a 20×20 image. Only the first 1000 faces from the face database, and the first 1000 non-faces from the non-face test database are used. This dataset is used because it can be argued that finding a representative set of non-face examples may be infeasible.

2. *Mfeat-Fou Digit4 (Mfeat)*: This is a dataset consisting of examples of ten handwritten digits, which can be found in [1]. In this dataset, Fourier components have been extracted from the original images, resulting in a 76-dimensional representation of each digit. 200 examples of each digit are available. In these experiments, digit *4* is used as the *target* class, and all the others as *outlier*.

3. *Geophysical (Geo)*: A multi-modal dataset, in which a *target* and *outlier* class are represented by spectra. In this problem, new *outlier* classes may appear during testing. 3982 *target* examples exist, and 3675 *outlier* examples.

Results

The results for a number of experiments on the real-world datasets are now presented. The objective of the experiments is to assess the SOCMC classifiers on the real-world problems to ascertain whether they do in fact outperform conventional discriminant-based classifiers. This paper also shows that the SOCMC classifiers can result in higher performance than the distance-based reject-option classifiers. In each experiment, SOCMC results are shown benchmarked against discriminant and reject-option classifiers. For a fair comparison, the same model and representation used for the discriminant classifier is used in the reject-option classifier, and also in the multi-class stage D_{MCC} of the SOCMC. A number of different D_{OCC} models are then chosen to attempt to improve the rejection performance $\text{perf}(\omega_t, \omega_r)$, with only the best results shown for brevity (there are examples where SOCMC classifiers do not work – some optimisation is required is to select appropriate models). The reject threshold for the reject-option and one-class classifiers is fixed to reject 5.00% of *target* examples for the given training set. As a starting point for the comparison, it is important to note that the SOCMC classifier results in a similar performance to the reject-option classifier when the same model (i.e. same representation and data model) is used for both the D_{OCC} and D_{MCC} results. Small differences in results are attributed to the fact that only a subset of $\mathbf{x}$ is used to train the D_{MCC}. These results are not included due to space constraints.

In Table 5.1 details of each experiment are shown. The first column indicates the dataset used, and the second column the model used for the discriminant classifier $\mathbf{M}$, the reject-option classifier $\mathbf{R}$ and the D_{MCC} stage of the SOCMC classifier $\mathbf{S}$. The last column shows the representation and classifier

Dataset	Base algorithm	SOCMC D_{OCC} model
Face A	PCA 0.99 QDC	PCA 0.99 Gauss
Face B	Fisher-map QDC	PCA 0.99 MoG-8
Face C	PCA 0.99 LDC	PCA 0.99 Gauss
Mfeat A	Nearest-mean	Gauss
Geo A	PCA 0.9 QDC	PCA 0.9 MoG-5
Geo B	PCA 0.999 QDC	PCA 0.999 MoG-5
Geo C	PCA 0.9 MoG-5/class	PCA 0.999 MoG-5

Table 5.1: Description of experiments. The first column shows the dataset used. In the second column the algorithm used for the discriminant classifier **M**, the reject-option classifier **R** and the D_{MCC} stage of the SOCMC classifier **S** is given. The last column shows the representation and classifier used for the D_{OCC} of the SOCMC. PCA is a principal component analysis mapping, followed by the percentage of retained variance. Gauss is a Gaussian model. MoG-N is a Mixture-of-Gaussians model with N mixtures. LDC and QDC are Bayes linear and quadratic classifiers respectively.

used for the D_{OCC} of the SOCMC. For each classifier, three performance results are shown (in terms of mean AUC[4] over 20-folds with standard deviations shown). These consist of the perf(ω_t, ω_o), perf$_{a1}(\omega_t, \omega_o)$ and perf$_{a2}(\omega_t, \omega_r)$ measures, denoted clf, $rj1$, and $rj2$ respectively. Ideally, all three performances should approach 1.00.

First we discuss the *face* results in Figure 5.5. In the first experiment *face A*, it can be seen that the discriminant classifier **MA** has a rejection performance ($rj1$ and $rj2$) that is much lower than the classification performance clf. This is attributed to the fact that the *target* decision space is unconstrained, providing little protection against changing conditions. The reject-option classifier **RA** then shows a marked improvement in rejection performance in terms of test $rj1$, with a small decrease in clf. This sacrifice of classification performance for improved rejection performance alludes to a tradeoff between these two measures. The poor performance on $rj2$ was unexpected at first, but on closer inspection of the model used (QDC), which assumes unimodality, and the fact that data generated in the $rj2$ test is also distributed in a uniform manner only in the region of the *target* class, may provide an adequate explanation. These results only show marginal (but significant at times) improvements of the SOCMC classifier over the reject-option. It is suspected that this dataset is largely unimodal, and close to Gaussian-distributed (and the outliers in $rj2$ are generated in a similar fashion). In the first experiment, the **SA** performances in terms of clf and $rj1$ are slightly better than **RA**. In the second experiment *face B*, **SB** results in a much higher rejection performance than **RB**, but with some loss in classification performance. Again we observe a trade-off between classification and rejection performance. The third experiment once again shows small improvements over the reject-option with respect to **SC**.

[4]where an ideal performance in a separable problem would result in an AUC score of 1.

In the left-most plot of Figure 5.6, the results of the *mfeat-fou digit4* experiments are shown. Here a nearest-mean classifier has been used, resulting in a 92.44% AUC classification performance for **MA**. The rejection performances are however around 50.00%. The reject-option classifier **RA** is not significantly better than **MA** at rejection. In this case a large number the outliers generated were accepted by a clearly sub-optimal rejection model, even though the classification performance is high. However the SOCMC classifier performs much better here. Even though a nearest-mean classifier is used for classification, the Gaussian model is much better at rejection. Low performances on $rj2$ suggest again that the *target* data is unimodal, with most generated *outlier* examples falling within the domain of the *target* class.

In the three right-most plots in Figure 5.6, the results of the *geophysical* experiments are shown, showing considerable improvements achieved by the SOCMC scheme. In *Geo A* it can be seen that both *RA* and *SA* improve in terms of $rj1$ performance. However the SOCMC is much better at $rj2$ performance. In this case, the D_{OCC} model used was a Mixture-of-Gaussians, that could model the apparent multi-modality of the *target* class, and thus provide better protection against the *outlier* examples generated in $rj2$. The reject-option rejection model was constrained to the unimodal QDC. In the second experiment *Geo B*, a good example of the SOCMC approach is shown (see **RB** and **SB**), with a clear performance improvement over the reject option. The third experiment shows that the SOCMC and reject option classifiers result in a similar performance, with a slightly better $rj2$ performance achieved by the SOCMC. We conclude that a strong classification model (fitting the data well)

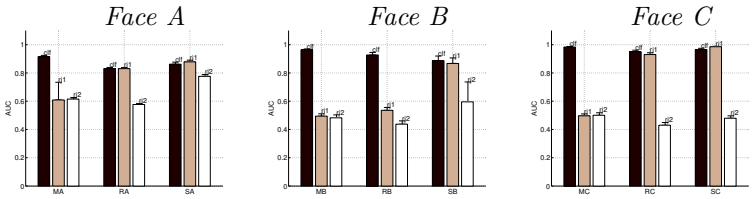


Figure 5.5: Summarised results of the *face-amsterdam* experiments.

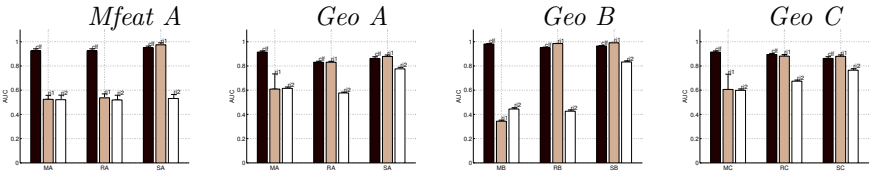


Figure 5.6: Summarised results of the *mfeat-fou digit4* and *geophysical* experiments.

will result in optimal classification and reject performance. It was observed that a discriminator can indeed obtain high classification performance, but a model chosen for good classification performance can be at the expense of

rejection performance. The SOCMC results showed that this classifier can improve upon the reject-option, with separate models trained locally for the purposes of classification and rejection respectively.

5.2.6 Conclusions

In this paper classification strategies for ill-defined problems was discussed. It was assumed that a well defined *target* class is to be discriminated from an ill-defined *outlier* class. First the implications on performance with respect to standard discrimination approaches was discussed, showing that a closed/constrained decision space around the *target* class is necessary for robustness to changing conditions. The state-of-the art classifier suited to this task is the distance-based reject option. It was pointed out that a practitioner should make use of an adequate evaluation methodology in selecting a classifier, considering both classification and rejection performance. A new classification strategy was proposed for these types of problems, involving the sequential combination of one-class and multi-class classifiers. These classifiers allow a model to be explicitly selected/trained in local regions of known overlap to emphasise either classification or rejection performance. Experimentation on a number of real-world datasets showed that in some cases the SOCMC classifier does indeed outperform the distance-based reject-option approach. An observation made during experimentation is that an inherent trade-off occurs between classification and rejection. Optimising this will be a focus of future research.

5.2.7 Acknowledgments

This research is/was supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs.

Bibliography

[1] Mfeat. *ftp://ftp.ics.uci.edu/pub/machine-learning-databases/mfeat/.*

[2] C.K. Chow. On optimum error and reject tradeoff. *IEEE Transactions on Information Theory*, It-16(1):41–46, 1970.

[3] K. Copsey and A. Webb. Classifier design for population and sensor drift. *Joint IAPR Workshops on Syntactical and Structural Pattern Recognition, and Statistical Pattern Recognition*, pages 744–752, August 2004.

[4] B. Dubuisson and M. Masson. A statistical decision rule with incomplete knowledge about classes. *Pattern Recognition*, 26(1):155–165, 1993.

[5] R.P.W. Duin. The combining classifier: To train or not to train? *in: R. Kasturi, D. Laurendeau, C. Suen (eds.), ICPR16, Proceedings 16th International Conference on Pattern Recognition (Quebec City, Canada), IEEE Computer Society Press, Los Alamitos*, 2:765–770, August 2002.

[6] C.L. Liu, H. Sako, and H. Fujisawa. Performance evaluation of pattern classifiers for handwritten character recognition. *International Journal on Document Analysis and Recognition*, pages 191–204, 2002.

[7] P. Paclík. Building road sign classifiers. *PhD thesis, CTU Prague, Czech Republic*, December 2004.

[8] T. V. Pham, M. Worring, and A. W. M. Smeulders. Face database. http://carol.wins.uva.nl/ vietp/publication/list.html.

[9] T. V. Pham, M. Worring, and A. W. M. Smeulders. Face detection by aggregated bayesian network classifiers. *Pattern Recognition Letters*, 23(4):451–461, February 2002.

[10] D.M.J. Tax. One-class classification. *PhD thesis, Delft Technical University, The Netherlands*, June 2001.

[11] D.M.J. Tax and R.P.W. Duin. Uniform object generation for optimizing one-class classifiers. *Journal for Machine Learning Research*, pages 155–173, 2001.

[12] A. Ypma, D.M.J. Tax, and R.P.W. Duin. Robust machine fault detection with independent component analysis and support vector data description. *Proceedings of the 1999 IEEE Workshop on Neural Networks for Signal Processing, Madison*, pages 67–76, 1999.

5.3 The interaction between classification and reject performance for distance-based reject-option classifiers

This section has been published as 'The interaction between classification and reject performance for distance-based reject-option classifiers', by T.C.W. Landgrebe, D.M.J. Tax, P. Paclík, and R.P.W. Duin, in *Pattern Recognition Letters, Special issue on ROC analysis*, **27(8)**:908-917,June 2006.

Abstract

Consider the class of problems in which a *target* class is well defined, and an *outlier* class is ill-defined. In these cases new *outlier* classes can appear, or the class-conditional distribution of the *outlier* class itself may be poorly sampled. A strategy to deal with this problem involves a two-stage classifier, in which one stage is designed to perform discrimination between known classes, and the other stage encloses known data to protect against changing conditions. The two-stages are, however, inter-related, implying that optimising one may compromise the other. In this paper the relation between the two stages is studied within an ROC analysis framework. We show how the operating characteristics can be used for both model selection, and in aiding in the choice of the reject threshold. An analytic study on a controlled experiment is performed, followed by some experiments on real-world data sets with the distance-based reject-option classifier.

5.3.1 Introduction

In pattern recognition, a typical assumption made during the design phase is that the various classes involved in a particular problem can be sampled reliably. However, in some problems, new classes or clusters may appear in the production phase that were not present during the design/training. In other problems, some classes may be sampled poorly, leading to inaccurate class models. Examples of applications that are affected by this are for instance:

- Diagnostic problems in which the objective of the classifier is to identify abnormal operation from normal operation [5]. It is often the case that a representative training set can be gathered for one of the classes, but due to the nature of the problem, the other class cannot be sampled in a representative manner. For example in machine fault diagnosis [20] a destructive test for all possible abnormal states may not be feasible or very expensive.

- Recognition systems that involve a rejection and classification stage, for example road sign classification. Here a classifier needs not only to discriminate between examples of road sign classes, but must also reject non-sign class examples [14]. Gathering a representative set of non-signs may not be possible. Similarly face detection [15], where a classifier must deal with well-defined face classes, and an ill-defined non-face class, and handwritten digit recognition [9], where non-digit examples are a serious issue.

For simplicity we consider the problem as one in which there is a well defined *target* class, and a poorly-defined *outlier* class. The primary objective is to maintain a high *classification* performance between known classes, and simultaneously to protect the classes of interest from new/unseen classes (or changes in expected conditions, reflected in the change of distribution of these

classes). We refer to the latter performance measure as *rejection* performance. Classification performance is defined between a well defined *target* class ω_t, and some partial knowledge existing for the *outlier* class ω_o. Rejection performance is defined between ω_t and a new (unseen) cluster/class from the *outlier* class ω_r that is not defined precisely in training.

Several strategies have been proposed. The first strategy to cope with this situation was proposed in [5], called the *distance*-based reject-option. Here a reject-rule was proposed to reject distant objects (with respect to the *target* class) post-classification. This evaluation differs considerably from the second strategy, the *ambiguity* reject option (defined in [5]) as proposed in [3]. In ambiguity reject a threshold is included to reject objects occurring in the *overlap* region between two known classes. There is assumed that all classes have been sampled in a representative manner. This is in contrast to this study, in which it is assumed that classes are poorly sampled or not sampled at all.

Classifiers with the reject rule differ from conventional classifiers in that two thresholds are used to specify the target area, namely a classification threshold θ, and a rejection threshold t_d (we define the target area to be the region in the feature space in which all examples are labelled *target*). A limitation of the distance-reject criterion is that the threshold itself has no direct relationship with the distribution of the known classes, as discussed in [13]. Thus a modified reject-rule was proposed in [13], involving computing the probability of a new object belonging to any of the known classes, based on covariance estimates. The threshold can then be based on on a degree of model-fit to the known classes.

In [8] we presented a third reject strategy, involving *combinations* of one-class [17] and supervised classifiers. This scheme allowed different models to be specifically designed for the purposes of classification or rejection. It was argued that a model optimised for the sake of classification may differ from that optimised for rejection, and that combining both optimised models can improve the overall combined classification/rejection performance. Experiments showed that this strategy outperforms the other reject-rules in some situations. It was also observed that a relation between the classification and rejection performance exists, and that optimising either performance is at the detriment of the other.

Each of the strategies has a classification and rejection threshold. In both [5] and [13], it has been shown how the distance-reject rule can be applied in practise, involving distance- or class conditional probability- thresholding of new incoming objects. In the case of the ambiguity-reject option, the classifiers can be evaluated and optimised since it is assumed that all classes have been sampled, as shown in [3] for known costs, and applied to imprecise environments in [7], and [19] to name a few. However, in the case of the distance-based reject option, a challenging problem posed is that the distribution of the unseen class is by definition absent, and thus standard cost-sensitive evaluations and optimisations become ill-defined, lacking a closed Bayesian formalism.

In [8], the ill-defined class problem was tackled by deriving strategies that can be used to study the way in which classification and rejection performance interact, based on the assumption that a new unseen class could occur anywhere in feature space. The rationale is that a minimal target area provides, in general, the most robust solution to an unseen class that could occur anywhere in feature space[5]. The methodology involved the artificial generation of the unseen class by assuming a uniformly distributed unseen class. Based on this methodology, it was observed that similar to the ambiguity-reject case, there is interaction between classification and rejection performance.

This paper is concerned with evaluating and optimising classifiers taking into account this interaction between classification and rejection. For this, Receiver-Operating Characteristic (ROC) curves will be used. ROC analysis [10], is a tool typically used in the evaluation of two-class classifiers in imprecise environments, plotting detection rate (true positive rate) against the false positive rate. We extend this analysis to the unseen class problem by including an additional dimension that is related to the general robustness of the classifier to an unseen class. A similar 3-dimensional ROC analysis has been applied elsewhere, such as in [7], [11], and [4], but in these cases this did not involve the ill-defined class problem. Our approach attempts to minimise the volume of the classes of interest in the feature space for robustness against unseen classes. It allows models to be compared (in a relative sense, since an absolute measure cannot be obtained) and provides insight into the choice of a reject-threshold that does not impact too much on classification performance.

In Section 5.3.2 an example is studied analytically to investigate the nature of the relation between classification and rejection rates, and the extended ROC analysis is presented. In Section 5.3.3 a criterion is proposed for the comparison of the extended ROC's. This criterion is applied to a synthetic 2-dimensional example with three different models. Finally, we discuss how to optimise an operating point (i.e. choose a classification and rejection threshold). Section 5.3.4 consists of a number of experiments to demonstrate the methodology in some realistic scenarios. Conclusions are given in Section 5.3.5.

5.3.2 The relation between classification and rejection performance

First we will develop our notation and illustrate the interaction between the classification and rejection performance by showing an example. In Figure 5.7 a synthetic example is presented in which ω_t and ω_o are two Gaussian-distributed classes distributed across domain x. Additionally we assume that a class ω_r is uniformly distributed across x. The class conditional densities for ω_t, ω_o and ω_r are denoted $p(x|\omega_t)$, $p(x|\omega_o)$, and $p(x|\omega_r)$ respectively, with priors $p(\omega_t)$, $p(\omega_o)$, and $p(\omega_r)$, which are assumed equal here. The unconditional density

[5]Rather than assuming that unseen classes can occur anywhere in feature space, it may be better to consider the nature of each problem, incorporating prior knowledge with respect to natural bounds in this space. To keep the discussion general, for now we assume a uniform, maximum entropy distribution.

$p(x)$ can then be written as in Equation 5.8. Note that in training we only have access to ω_t and ω_o, and in testing ω_r will also appear.

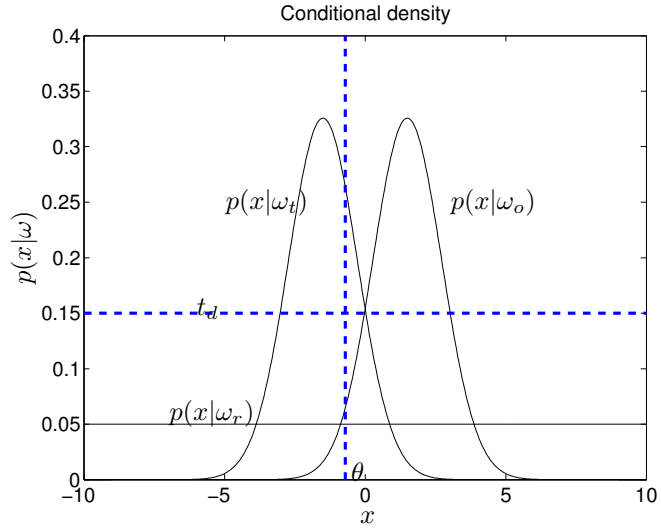


Figure 5.7: A synthetic example, illustrating the class conditional densities for ω_t, ω_o, and ω_r, with a distance-based reject-option classifier. The classification boundary is specified by θ, and the rejection boundary by t_d, $t_d = 0.15$.

For the total probability distribution of x therefore holds:

$$p(x) = p(\omega_t)p(x|\omega_t) + p(\omega_o)p(x|\omega_o) + p(\omega_r)p(x|\omega_r) \tag{5.8}$$

For this 1-dimensional data, a classifier is defined which only consists of a single threshold, denoted θ. The position of θ determines the classification performance, and can be specified given a desired true-positive rate (TP_r) or false-positive rate (FP_r). As θ varies, so do the respective TP_r and FP_r, resulting in the ROC between ω_t and ω_o. In a typical discrimination problem (ignoring the reject threshold), we can define the true positive rate (TP_r) and false positive rate (FP_r) in terms of θ as in Equation 5.9.

$$\begin{array}{rcl} TP_r(\theta) & = & \int_{-\infty}^{\infty} p(\omega_t)p(x|\omega_t)I(x|\theta)dx \\ FP_r(\theta) & = & \int_{-\infty}^{\infty} p(\omega_o)p(x|\omega_o)I(x|\theta)dx \end{array} \tag{5.9}$$

The indicator function $I(x|\theta)$ specifies the relevant domain, as defined in Equation 5.10.

$$I(\mathbf{x}|\theta) = \begin{cases} 1 \text{ if } p(\omega_t)p(\mathbf{x}|\omega_t) - p(\omega_o)p(\mathbf{x}|\omega_o) > \theta \\ 0 \text{ otherwise} \end{cases} \tag{5.10}$$

In a typical discrimination problem, the evaluation criterion used is the mean classification error between ω_t and ω_o and the ability to reject new unseen

classes ω_r is often not considered. In ill-defined classification problems, the performance with respect to both the known outliers ω_o and the unknown outliers ω_r is important [8]. The $FP_r(\theta)$ can therefore be decomposed into two parts:

$$\begin{array}{rcl} FP_r^o(\theta) & = & \int_{-\infty}^{\infty} p(x|\omega_o) I(x|\theta) dx \\ FP_r^r(\theta) & = & \int_{-\infty}^{\infty} p(x|\omega_r) I(x|\theta) dx \end{array} \tag{5.11}$$

Standard classifiers will ignore FP_r^r and only focus on minimising FP_r^o. For these situations the distance-based rejection option classifier [5], or the combined sequential one-class and multi-class classifier [8] should be used. A reject-option classifier is demonstrated in Figure 5.7.

It can be seen that the class conditional density $p(x|\omega_t)$ is thresholded such that any object x assigned to ω_t will only be accepted if $p(x|\omega_t) > t_d$. Thus for the distance-based reject-option classifier, TP_r (Equation 5.9), FP_r^o and FP_r^r (Equation 5.11), can be written for the general multivariate case as:

$$\begin{array}{rcl} TP_r(\theta, t_d) & = & \int_{-\infty}^{\infty} p(\omega_t) p(\mathbf{x}|\omega_t) I(\mathbf{x}|t_d, \theta) d\mathbf{x} \\ FP_r^o(\theta, t_d) & = & \int_{-\infty}^{\infty} p(\omega_o) p(\mathbf{x}|\omega_o) I(\mathbf{x}|t_d, \theta) d\mathbf{x} \\ FP_r^r(\theta, t_d) & = & \int_{-\infty}^{\infty} p(\omega_r) p(\mathbf{x}|\omega_r) I(\mathbf{x}|t_d, \theta) d\mathbf{x} \end{array} \tag{5.12}$$

where $I(\mathbf{x}|t_d, \theta)$ is the indicator function:

$$I(\mathbf{x}|t_d, \theta) = \begin{cases} 1 \text{ if } p(\mathbf{x}|\omega_t) > t_d \textit{ and } p(\omega_t) p(\mathbf{x}|\omega_t) - p(\omega_o) p(\mathbf{x}|\omega_o) > \theta \\ 0 \text{ otherwise} \end{cases} \tag{5.13}$$

Equation 5.12 can be used to study the complete operating characteristic of a classifier. Note that in a real situation, it is unlikely that the class-conditional densities are known. A typical classifier evaluation in these situations involves computing the ROC curve on an independent test set.

We define the complete operating characteristic by all combinations of θ and t_d. The operating characteristic of the example in Figure 5.7 is illustrated in Figure 5.8. This is similar to standard ROC analysis, in which the TP_r is traded off against the FP_r. Here the FP_r is decomposed into FP_r^o and FP_r^r, resulting in a 3D ROC plot with a 2D ROC *surface*.

In Figure 5.8, the operating characteristics are shown for a number of rejection thresholds (t_d), and across all classification thresholds (θ). The left-column plots depict the class conditional density distributions, with the respective t_d shown in relation to the actual ω_r distribution. In the centre column plots, the ROC curve is presented for all TP_r, FP_r^o, and FP_r^r. By projecting this on the (TP_r, FP_r^o)-plane, the traditional ROC curve is retrieved. The plots in the right column simplify the 3-dimensional surface, plotting the classification performance in terms of mean classification performance for each θ and t_d, against FP_r^r. The mean classification performance is defined as

$$p_{mean}(\theta, t_d) = 1 - \frac{(1 - TP_r(\theta, t_d)) + FP_r^o(\theta, t_d)}{2} \tag{5.14}$$

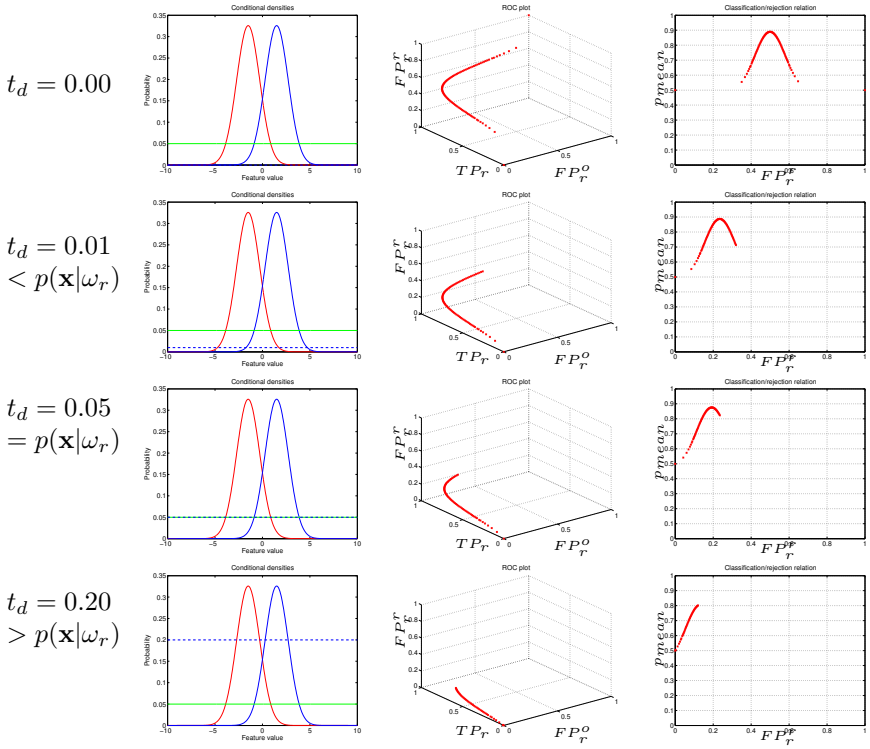


Figure 5.8: Fixed t_d, and varying θ.

It can be observed that as t_d is increased, the FP_r^r progressively decreases: the amount of unseen data that is classified as *target* decreases, and consequently the rejection performance increases. In the top row ($t_d = 0$) there is no rejection protection, the classification performance is maximal but the rejection performance is very poor (high FP_r^r). As soon as some t_d is enforced, the FP_r^r decreases radically, indicating a lower probability of accepting a randomly distributed example from ω_r. Even a very loose boundary in the tails of the ω_t distribution significantly decreases the volume of the decision space. As t_d is increased, the rejection performance increases at some sacrifice of classification performance. This effect is most apparent when t_d is quite high, due to the fact that more *target* examples from the tails of the distribution are excluded.

This synthetic example makes it evident that classification and rejection are inter-related. In Section 5.3.3, these extended ROC plots are used to derived a performance criteria to evaluate different models in these situations, and provide insight into threshold selection.

5.3.3 Model selection and optimisation

Now a model selection criterion is formalised that makes use of the full operating characteristics, extending ROC analysis to this problem domain. This will be developed and demonstrated by a synthetic example using three different classifier models.

Model selection

To select the optimal model, an evaluation criterion for the 3D ROC should be defined. In [11], [4], and [7] the 3-dimensional *volume* under the 2-dimensional surface is related to the overall performance. This is an extension of the AUC (Area under the ROC curve) [2]. In a similar manner, we derive an evaluation of the 3-dimensional ROC in our analysis, providing a measure of the classification-rejection performance for the classifier. The ROC in this case plots the TP_r achieved against FP_r^o and FP_r^m (see the example in Figure 5.8). Plotting $1 - TP_r$ against FP_r^o and FP_r^m, it is evident that the volume under this ROC surface should be minimal in the ideal case, implying generally that the classifier achieves low classification error rates, and has high rejection performance (low volume of *target* decision space). Formalising this performance criterion, we derive the VUC (Volume under the ROC):

$$VUC(\theta, t_d) \quad = \quad 1 - \int\int (1 - TP_r(\theta, t_d)) dFP_r^o(\theta, t_d) dFP_r^r(\theta, t_d) \qquad (5.15)$$

The volume itself is subtracted from 1 to form a performance measure (high scores are favourable). In some cases it may be sensible to also integrate over a restricted range of $FP_r^o(\theta, t_d)$ (by restricting the range of θ). This effectively analyses a patch of the ROC surface. This is because an AUC integrated over all classification decision thresholds is not always ideal. This occurs in the case in which the ROC surfaces of two different models intersect. In this case model selection is operating-point dependent [1], and thus the ROC should only be analysed for a range of interest.

Evaluation on artificial data

In Figure 5.9 a scatter-plot of the data is shown. The left plot shows the data available in training (ω_t and ω_o only), and the right plot shows an additional reject-class uniformly distributed, assumed to occur during testing. The dataset consists of 1600 ω_t examples, 800 ω_o examples, and 2400 ω_r examples. The experimental procedure involves using 80% of the data for training, and 20% for testing, with ω_r excluded from training. This is repeated 10 times, following a randomised hold-out procedure. The right plot also depicts the decision boundary of three different classifiers trained on the data, namely a Bayes linear classifier (LDC), a Bayes quadratic classifier (QDC), and a Mixture of Gaussians classifier (MOGC), with three clusters per class. The decision boundary is plotted for a single fixed classification and rejection threshold. From the decision boundaries, it is clear that the MOGC fits the data well, as opposed

to the LDC and QDC models. These weaker models are typical in real-high dimensional problems where both data and computation time are limited, and thus a model choice may not be obvious.

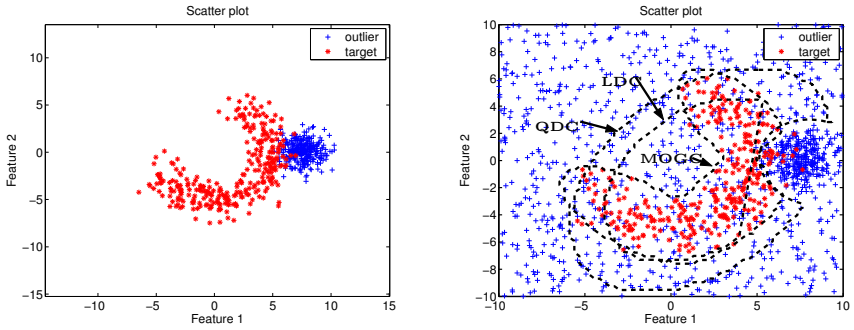


Figure 5.9: Scatter-plot for the synthetic example. The left plot shows the data available in training, and the right plot shows a testing scenario, in which a new unseen class exists that should be rejected. The right plot also shows the classifier decision boundaries for three classifiers, at a set operating point. The classifiers are labelled LDC, QDC and MOGC respectively.

In Figure 5.10 the operating characteristics are shown for the three classifiers. The top row shows the full operating characteristics, and the bottom row depicts the mean classification performance (see Equation 5.14) versus the FP_r^r. These clearly show that the classification-rejection characteristic varies considerably across the different models. The MOGC is able to achieve a higher mean classification performance for a lower FP_r^r than the LDC and QDC for most regions. For example, for MOGC, it can be seen that the optimal p_{mean} is over 90.00% for a FP_r^r of around 25.00% (based on the assumed ω_r distribution). The LDC, however, only achieves a p_{mean} of around 80.00% for the same degree of rejection performance. Even though the ω_r distribution is unknown, it is apparent in this case that the MOGC classifier can achieve higher rejection robustness than the LDC for the given classification performance. A lower FP_r^r is indicative of a reduced *target* decision space, and the model in question is thus less likely to accept a randomly distributed example from ω_r.

In the lower plots, the top-right characteristics/curves correspond to the case in which $t_d = 0.0$, with curves corresponding to increasing t_d shown to the left of this. A general observation that can be made is that for low values of t_d, the classification performance decreases rapidly for increasing t_d, and the FPr^r decreases (improving rejection performance). This was expected, showing once again that the classification and rejection performances should be traded off, but also that for large t_d, both classification and rejection performances decrease rapidly. It is also clear that only some (typically low) values of t_d make practical sense, since a large value leads to a very poor recovery of ω_t examples.

In Table 5.2, the VUC is computed for the synthetic problem for each fold. This performance measure indicates that the MOGC classifier is superior,

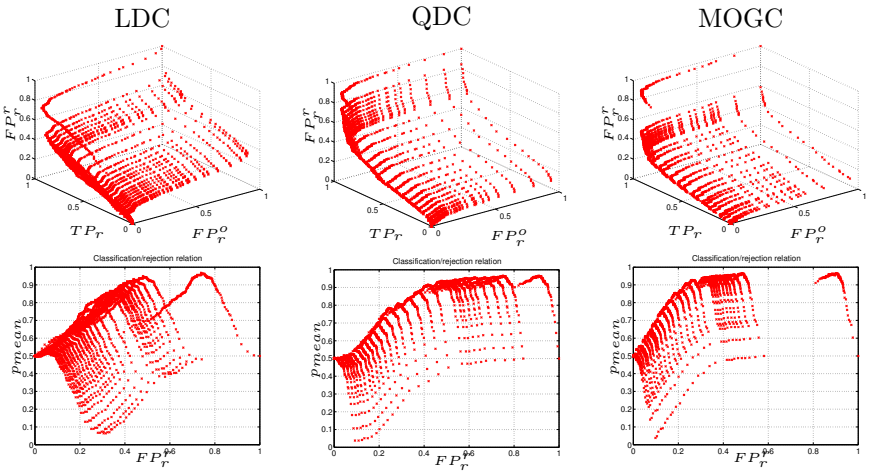


Figure 5.10: Operating characteristics for the three classifiers. The left column consists of LDC results, followed by QDC results in the centre column, and MOGC results in the right column. The top row depicts the full ROC, i.e. all possible operating points. The bottom row shows the mean classification performance versus FP_r^r, clearly showing the interplay between the two measures.

significantly better than the QDC and LDC model. This is an expected result, since the decision boundary fits the class distribution well, providing both a high classification performance, and a lower decision space volume for high rejection performance. The QDC model is superior to the LDC model. Thus the VUC measure proved to be a useful performance criterion, sensitive to both classification and rejection capabilities of the classifier. Note that in this example, computing the error rate only (on known data) results in competitive performance between QDC and MOGC, however the new VUC criterion and the evaluation methodology showed that MOGC is in fact the better choice since it is better at rejection. Table 5.2 also shows the results of two other experiments. The ϵ_{rej} measure shows the performance of the three reject-option classifiers for a chosen set of thresholds ($\theta = minimum\,error\,point, t_d = 0.05$) based on 10 independent sets of data (drawn from the same distribution, with 1600 ω_t examples, 800 ω_o examples, and 3200 uniform ω_r examples), in which the classifiers were trained on the original set. This performance measure is a simple error rate measure that averages the ω_t and other errors:

$$\epsilon_{rej} = \frac{1}{3}((1 - TP_r) + FP_r^o + FP_r^r) \tag{5.16}$$

These tests confirm the results of the VUC model selection i.e. that the MOGC is the most appropriate choice. In order to demonstrate the advantage of using a reject option, this same test is repeated on the three same models without reject option, resulting in ϵ_{norej}. In each case it is clear that performance is

	VUC	ϵ_{rej}	ϵ_{norej}
LDC	0.846 ± 0.005	0.197 ± 0.009	0.353 ± 0.007
QDC	0.907 ± 0.003	0.116 ± 0.006	0.420 ± 0.004
MOGC	0.928 ± 0.005	0.087 ± 0.004	0.404 ± 0.003

Table 5.2: Comparing the 3 classifiers in the case study. VUC is computed for each classifier (high scores are favourable), serving as the model selection criterion. The score ϵ_{rej} is the classification error obtained on the independent tests with fixed thresholds for the classifiers with reject option, and ϵ_{norej} is the error on the same models without reject protection.

significantly worse compared to the classifiers with reject-option. Interestingly, the MOGC classifier without reject-option fared worse than the QDC model since it optimised by surrounding the ω_o class, resulting in a very large decision space i.e. poor rejection performance.

Choosing an operating point

Based on the VUC measure, the most appropriate model (on average) was selected. In the example, the MOGC classifier was found to be superior, and this was demonstrated further by computing error-rates on independent data at a specific operating point. Subsequent to the model selection, the next step is to choose classification and rejection thresholds best suited to the problem. In the standard cost-sensitive approach [16], this would involve minimising the overall system loss L, given costs and priors. Assume c_t is the cost of misclassifying a ω_t example, c_o the cost of misclassifying a ω_o example, and c_r the cost of misclassifying an ω_r example (ignoring to which class an error is assigned). The loss can then be computed as:

$$L = \theta p(\omega_t) c_t (1 - TP_r) + (1 - \theta) p(\omega_o) c_o FP_r^o + t_d p(\omega_r) FP_r^r \tag{5.17}$$

Minimising L involves computing L for all combinations of θ and t_d until the optimal thresholds are found (or geometrically, intersecting an iso-performance *surface* on the ROC surface). However in this problem, since the distribution of ω_r is unknown, the concept of optimality becomes undefined (true FP_r^r values cannot be obtained). The synthetic experiments have, however, shown how to analyse the impact of a varying t_d on classification performance, and decision space volume. Thus a practical step that can be taken in aiding optimisation is to attempt to choose a t_d such that the ω_t decision space is minimised, without sacrificing too much classification performance. Given this premise, we propose that the classification threshold should be optimised first. This is because we have sampled these classes properly, allowing for a cost-sensitive design. Once θ has been chosen, a t_d should be selected that encloses ω_t. The operating characteristics can also be helpful in inspecting the sensitivity over a range of t_d, where a less sensitive choice is to be preferred.

Summary

In summary, the following steps are involved in generating the full operating characteristics, given a model D, a *target* class ω_t, and an *outlier* class ω_o:

- Assume a distribution for ω_r, for example uniformly distributed around ω_t, and generate data accordingly.
- Train D using ω_t and ω_o.
- Define a range of classification and rejection thresholds θ and t_d. For each threshold, compute the respective TP_r, FP_r^o, and FP_r^r, using independent sets of ω_t and ω_o, and the ω_r data.

The performance criterion VUC can then be used to perform model selection, and the thresholds can be chosen using the operating characteristics.

5.3.4 Experiments

In this section a number of real-world examples are conducted, demonstrating practical application of the proposed ROC analysis methodology. Model selection criteria are compared for a number of competing models, and the performance of a classifier with reject-option is compared to the same model, without reject-option. In each case, an independent test set is applied, in which the ω_r class is unseen in training, simulating the effect an unseen class may have on each classifier. These validation tests are performed to demonstrate the applicability of the VUC measure in these ill-defined classification problems.

The following datasets have been used for these experiments, in which the objective is to detect *target* examples as well as possible, without accepting too many examples from ω_o or ω_r:

1. *Phoneme*: This dataset is sourced from the ELENA project [6], in which the task is to distinguish between oral and nasal sounds, based on five coefficients (representing harmonics) of cochlear spectra. In this problem, the "nasal" class is chosen as ω_t. A $k-$means clustering is performed on the "oral" class, requesting three clusters. The first two clusters are are regarded as ω_o (used in training), and the third cluster is treated as ω_r (not used in training).

2. *Mfeat*: This is a dataset consisting of examples of ten handwritten digits, originating from Dutch utility maps[6]. In this dataset, Fourier components have been extracted from the original images, resulting in a 76-dimensional representation of each digit. 200 examples of each digit are available. In these experiments, digits *2* and *6* are used as the *target* class, digits *4*, *5*, *6* and *8* as ω_o, and digits *1*, *2* and *9* as ω_r (not used in training).

[6] Available at ftp://ftp.ics.uci.edu/pub/machine-learning-databases/mfeat/

3. *Satellite*: This dataset consists of 6435 multi-spectral values of a satellite image [12], with 36 dimensions (4 spectral bands in a 9 pixel neighbourhood). Six classes have been identified to characterise the topography, labelling the dataset accordingly. In experiments the fifth class is considered ω_t , classes 1, 4, and 6 are considered known ω_o classes, and classes 2 and 3 are unseen in training (ω_r).

The following procedure is used in each experiment (similar to the case study presented in Section 5.3.3):

- An independent test set (ω_t examples only) is extracted from the original data $\mathbf{x}$ (containing examples from ω_t and ω_o), resulting in $\mathbf{x}_{val1}$, with the remainder called $\mathbf{x}_{tr}$. In the experiments, 50% of the *target* examples are extracted.

- Various clusters/sub-classes of the ω_o class in $\mathbf{x}$ are extracted, resulting in $\mathbf{x}_{val2}$, with $\mathbf{x}'_{tr}$ remaining . It is important to note that $\mathbf{x}_{val2}$ now contains data that will not be used to train the classifier, but will be applied only in the validation test. Since these extracted classes may have a very different distribution to that of $\mathbf{x}'_{tr}$, a classifier with reject-option is expected to result in better performance.

- $\mathbf{x}_{val1}$ and $\mathbf{x}_{val2}$ are combined into a single validation set, $\mathbf{x}_{val}$.

- The VUC for each model is estimated following a 10-fold randomised hold-out procedure, utilising the $\mathbf{x}'_{tr}$ dataset, and an assumed uniform distribution of ω_r. Since in these ill-defined problems the ω_r class is absent (by definition), we assume that examples of this class can occur randomly in feature space (a worst-case scenario). Thus an additional class is generated artificially such that this new data surrounds the ω_t class uniformly. For efficiency reasons in high-dimensional problems, the data is generated in a hyper-sphere rather than a hyper-cube [18]. Additionally the data is generated in a subspace of the *target* class, within a PCA (Principal Component Analysis) subspace, retaining 99.9% of variance. This effectively results in the generation of new objects (the new examples can be reprojected into the original space using the inverse of the PCA mapping). The original data is scaled to unit variance, and the artificial data is then generated within this space with a radius of 1.1 of the covariance of ω_t.

- Following the VUC estimation, the classifier is trained using the full $\mathbf{x}'_{tr}$ data. The classification threshold is optimised to the equal error point, and an appropriate reject threshold is chosen for each dataset according to the operating characteristics (obtained in the previous step). The same data is then used to train a second classifier, using the same model, but without reject option. The validation set $\mathbf{x}_{val}$ is then applied to each classifier, and the respective error-rates of each classifier are computed, denoted ϵ_{rej} for the reject-option classifier, and ϵ_{norej} for the classifier

without reject-option. The mean percentage difference in classification error-rates between a classifier with and without reject-option is then computed ($100mean(\epsilon_{rej} - \epsilon_{norej})$). It is expected that the reject-option classifier should result in improved performance if the independent $\mathbf{x}_{val2}$ data distribution varies considerably from the trained distribution.

Thus the objectives of the experiments are to validate the usage of the performance criterion in these realistic scenarios, and show cases in which a good model choice, and reasonable choice of thresholds results in a performance improvement. Note that real data only is used in the validation step, resulting in a realistic set of experiments.

In Table 5.3, the experimental results are presented. Three groups of results are shown, corresponding to the *Phoneme*, *Mfeat* and *Satellite* experiments respectively. The first column describes the classifier used, as well as the respective feature extraction procedure, and the rejection threshold used for the validation test. The second column gives the VUC results for each model (with standard deviation shown over 10 folds), in which high scores are favourable. In the third column, the reject-option classifier error rate is compared to the same classifier without reject-option. The error rate percentage of the classifier without reject-option is shown subtracted from the error rate percentage of the classifier with reject-option. A positive result here indicates the percentage improvement (and a negative value indicates the reject-option classifier has performed worse). Note that the test set here consists of both an independent test set from ω_t, and an unseen class/cluster, and thus this measure gives an overall impression of the classification-rejection performance improvement.

In the *Phoneme* results it can be seen that four of the five models result in a large performance improvement in the independent tests. The PCA-3D QDC model does not result in improved performance, which is also supported by a lower VUC. The QDC model performs much better, indicating that dimensionality reduction is not appropriate here (the original data only has 5 features). This argument is also strengthened by the higher VUC measure. In the three Mixture-of-Gaussian tests, it seems apparent that a lower number of mixtures results in better performance. The *Mfeat* experiments consider four different models following a Fisher dimensionality reduction. In the case of the NMC (Nearest Mean Classifier), it should be mentioned that small values of t_d results in no rejection at all, and only very large values result in rejection protection. However, at this point, the impact on classification performance is severe, and thus the model is unsuitable for this task. This is also suggested by a lower VUC score. The three other models result in comparable performance. In the *Satellite* experiments, the NMC results in a similar situation to that seen in the *Mfeat* case. Both the Fisher LDC, and the PCA-4D LDC models result in a similar performance in the validation test. However it can be seen that the Fisher LDC has a significantly larger VUC. This discrepancy may be due to the fact that the VUC averages performance over all possible operating points, and is thus not locally sensitive. In this case it may make more sense to integrate over a smaller range (given

that some prior knowledge about the problem exists). In these two cases, a more local VUC was performed, denoted VUC_2, in which the integration was applied over the full range of θ, and over a restricted t_d range, $0.0 < t_d \leq 0.2$. These experiments resulted in $VUC_2(Fisher, LDC) = 0.591 \pm 0.0111$, and $VUC_2(PCA\,4D, LDC) = 0.554 \pm 0.012$. It is clear that the performance measures are now more similar, which is an expected result[7].

On the whole, the experiments show that the derived VUC measure is useful in identifying more appropriate models. An important observation that can be made is that a reject-option classifier does not always result in adequate/beneficial protection against unseen classes. The ROC analysis approach presented here helps to identify these cases. Another point that should be raised is that an adequate rejection threshold varies according to the problem and model. The implication is that a reject-threshold setting that does not consider the operating characteristics may have little or a detrimental effect on performance. This is an important consideration that is highlighted and dealt with in this paper.

5.3.5 Conclusion

Classifiers designed to protect a well-defined *target* class from ill-defined conditions, such as new unseen classes, are defined by two decision thresholds, namely a classification and rejection threshold. The classification threshold is designed to provide an optimal trade-off between known classes, and the rejection threshold protects the *target* class against changes in conditions e.g. new unseen classes.

In this paper, we discussed the fact that classification and rejection performances are not independent, but that there is an interplay between them. The consequence of the interplay is that independently optimising classification performance may be at the expense of rejection performance, and the opposite also holds. Even though this interaction is expected, the fact that the unseen class is absent makes it difficult to devise a model selection and optimisation strategy that results in a classifier with both good classification and rejection performance. This paper tackled this problem by measuring how well the classifier protects the *target* class from a uniformly distributed ill-defined class, effectively resulting in a measure proportional to the volume occupied by the *target* class decision space. This measure aids in choosing and optimising a classifier that reduces the risk of misclassifying an unseen class (without too much loss of classification performance) since we can now inspect both the classification performance, and volume of the decision space.

The investigation of this problem involved the extension of classical 2-dimensional ROC analysis by including the errors associated with the unseen class as an additional dimension of the ROC. This results in a 3-dimensional ROC surface, allowing the classification-rejection dynamics to be investigated.

[7]Note that since a limited range is used, this performance measure is not bound between 0 and 1, but is instead bound by the volume over which the integration is performed in the unit cube.

Experiment	VUC	Percentage gain
Phoneme		
QDC $t_d = 0.01$	0.742 ± 0.011	21.67 ± 0.40
PCA-3D QDC $t_d = 0.01$	0.553 ± 0.009	-0.19 ± 0.05
MOGC 1 4 $t_d = 0.01$	0.667 ± 0.036	19.80 ± 5.60
MOGC 1 3 $t_d = 0.01$	0.678 ± 0.022	19.26 ± 6.21
MOGC 1 2 $t_d = 0.01$	0.708 ± 0.021	22.10 ± 4.03
Mfeat		
Fisher NMC $t_d = 0.70$	0.341 ± 0.027	-1.81 ± 0.72
Fisher LDC $t_d = 0.05$	0.503 ± 0.034	28.49 ± 1.20
Fisher QDC $t_d = 0.05$	0.504 ± 0.032	29.22 ± 1.01
Fisher MOGC 2 3 $t_d = 0.05$	0.504 ± 0.027	29.95 ± 0.73
Satellite		
Fisher NMC $t_d = 0.70$	0.374 ± 0.0188	-1.58 ± 0.25
Fisher LDC $t_d = 0.10$	0.612 ± 0.009	10.27 ± 0.26
PCA-4D LDC $t_d = 0.10$	0.489 ± 0.015	12.10 ± 0.34

Table 5.3: Summary of experimental results on the *Phoneme*, *Mfeat*, and *Satellite* datasets, comparing models for the VUC performance criterion (high scores are favourable), and comparing the mean absolute percentage difference in classification error-rates between a classifier with and without reject-option on independent test sets. The standard deviations are also shown over the 10-fold experiments. PCA is a principal-component analysis representation, followed by the number of retained components, and Fisher is a Fisher-projection to 1-dimension. NMC is a nearest-mean classifier, LDC is a Bayes linear classifier, QDC is a Bayes quadratic classifier, and MOGC is a Mixture of Gaussians classifier followed by the numbers of mixtures used per class.

This was demonstrated via a simple analytic example, and subsequently used to devise a performance measure involving integrating the volume of the ROC plot, resulting in the Volume under the ROC (VUC), which is analogous to the Area under the ROC measure. Experiments were performed which showed the effectiveness of this measure in selecting the most appropriate model for the problem. Real experiments validated the measure by including a test involving real unseen classes/clusters, in which there was a consistency between good VUC scores and classifier performance with respect to the unseen data. The experiments made it clear that careful attention should be paid in the choice of the reject threshold, showing how the proposed ROC analysis can lead to a solution involving minimal impact on classification performance, but large impact on reducing the risk of accepting unseen class examples.

5.3.6 Acknowledgements

This research is/was supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of

Economic Affairs. A special mention is given to the anonymous reviewers who helped clarify some aspects of this work.

Bibliography

[1] N.M. Adams and D.J. Hand. Comparing classifiers when misallocation costs are uncertain. *Pattern Recognition*, 32(7):1139–1147, 1999.

[2] A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[3] C.K. Chow. On optimum error and reject tradeoff. *IEEE Transactions on Information Theory*, It-16(1):41–46, 1970.

[4] S. Dreisetl, S. Ohno-Machado, and M. Binder. Comparing trichotomous tests by three-way ROC analysis. *Medical Decision Making*, 20(3):323–331, 2000.

[5] B. Dubuisson and M. Masson. A statistical decision rule with incomplete knowledge about classes. *Pattern Recognition*, 26(1):155–165, 1993.

[6] ELENA. European ESPRIT 5516 project. *phoneme dataset*, 2004.

[7] C. Ferri and J. Hernandez-Orallo. Cautious classifiers. *First workshop on ROC analysis in AI (ROCAI-2004)*, pages 27–36, August 2004.

[8] T.C.W. Landgrebe, D.M.J. Tax, P. Paclík, R.P.W Duin, and C.M. Andrew. A combining strategy for ill-defined problems. *Fifteenth Annual Symposium of the Pattern Recognition Association of South Africa*, pages 57–62, November 2004.

[9] C.L. Liu, H. Sako, and H. Fujisawa. Performance evaluation of pattern classifiers for handwritten character recognition. *International Journal on Document Analysis and Recognition*, pages 191–204, 2002.

[10] C. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, 3(4), 1978.

[11] D. Mossman. Three-way roc's. *Medical Decision Making*, 19:78–89, 1999.

[12] P.M. Murphy and D.W. Aha. UCI repository of machine learning databases, ftp://ftp.ics.uci.edu/pub/machine-learning-databases. *University of California, Department of Information and Computer Science*, 1992.

[13] R Muzzolini, Y.H. Yang, and R. Pierson. Classifier design with incomplete knowledge. *Pattern Recognition*, 31(4):345–369, 1978.

[14] P. Paclík. Building road sign classifiers. *PhD thesis, CTU Prague, Czech Republic*, December 2004.

[15] T. V. Pham, M. Worring, and A. W. M. Smeulders. Face detection by aggregated bayesian network classifiers. *Pattern Recognition Letters*, 23(4):451–461, February 2002.

[16] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42:203–231, 2001.

[17] D.M.J. Tax. One-class classification. *PhD thesis, Delft Technical University, The Netherlands*, June 2001.

[18] D.M.J. Tax and R.P.W. Duin. Uniform object generation for optimizing one-class classifiers. *Journal for Machine Learning Research*, pages 155–173, 2001.

[19] F. Tortorella. Reducing the classification cost of support vector classifiers through an ROC-based reject rule. *Pattern Anal. Applic.*, 7:128–143, 2004.

[20] A. Ypma, D.M.J. Tax, and R.P.W. Duin. Robust machine fault detection with independent component analysis and support vector data description. *Proceedings of the 1999 IEEE Workshop on Neural Networks for Signal Processing, Madison*, pages 67–76, 1999.

Chapter 6

Multi-stage classification systems

6.1 Overview

This chapter investigates the use of operating characteristics for multi-stage classifier system design. These classifier systems comprise a number of stages that can be connected in series or parallel[1]. Operating characteristics can be useful in this application because there may be interactions between the various stages that can be accounted for. The operating characteristics can thus be used to *holistically* design the system, and set thresholds that account for both inter-class and inter-stage interactions.

Only a special case is considered in this chapter, consisting of a typical topology used in recognition systems. Two stages are combined in series, with the first stage performing the task of detection, recovering 'target' class objects from a typically widely distributed 'non-target' class. For example, in road-sign recognition [1], road-sign images are detected in images in the first stage. The second stage then consists of a supervised classifier that distinguishes between the various 'target' classes.

There is scope for much further research in this area, generalising the operating-characteristic approach to other multi-stage systems. Larger systems would have similar computational restrictions to those raised in chapter 4. One interesting opportunity would be to make use of the decomposition approach in the second part of chapter 4 to simplify the multi-stage operating characteristics.

[1] Classifiers such as boosting algorithms [2] are also multi-stage systems, but would not be considered here because they are lower-level components of a single classifier. Of interest in this chapter are combinations of independent trained classification systems.

Bibliography

[1] P. Paclík. Building road sign classifiers. *PhD thesis, CTU Prague, Czech Republic*, December 2004.

[2] P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. *Neural Information Processing Systems*, 14, December 2001.

6.2 Optimising two-stage recognition systems

This section has been published as 'Optimising two-stage recognition systems', by T.C.W. Landgrebe, P. Paclík, D.M.J. Tax, and R.P.W. Duin, in *International workshop on multiple classifier systems, Monterey California, USA*, June 2005.

Abstract

A typical recognition system consists of a sequential combination of two experts, called a detector and classifier respectively. The two stages are usually designed independently, but we show that this may be suboptimal due to interaction between the stages. In this paper we consider the two stages holistically, as components of a multiple classifier system. This allows for an optimal design that accounts for such interaction. An ROC-based analysis is developed that facilitates the study of the inter-stage interaction, and an analytic example is then used to compare independently designing each stage to a holistically optimised system, based on cost. The benefit of the proposed analysis is demonstrated practically via a number of experiments. The extension to any number of classes is discussed, highlighting the computational challenges, as well as its application in an imprecise environment.

6.2.1 Introduction

In this paper we view the sequential combination of two classifiers as a Multiple Classifier System (MCS). We illustrate that the independent design of individual classifiers in such sequential systems results in sub-optimal performance, since it ignores the interaction between stages. In this paper we demonstrate that optimality can be obtained by viewing such an MCS in a holistic manner. This research is targeted specifically at two-stage recognition systems, in which the first stage classifier attempts to detect *target* object distributed among a typically poorly sampled, or widely distributed *outlier* class. The second classifier then operates on objects selected by the first, and discriminates between sub-*target* classes. An example is image-based road-sign recognition [9], in which the first stage involves detecting road-signs that are distributed among an arbitrary background, and the second stage consists of a classifier to distinguish between different sign classes. Another application is fault diagnosis, such as [7], in which the first stage classifier is designed to detect a fault from normal operation, and the second stage to characterise the type of fault.

Considering the detector, since the *outlier* class is poorly defined, a two-class discrimination scheme is inappropriate, and other methods that are trained / designed only on the *target* class are typically used, such as correlation. Recently One Class Classification (OCC) was introduced [12], consisting of a formal framework to train models in situations in which data from only a single class is available. This allows a statistical pattern recognition methodology to be taken in designing the detector[2]. Thus we consider these recognition systems as a mixture of one-class and multi-class classifiers.

[2]Note that the MCS view on such a multi-stage system also holds for two-stage recognition systems that are constructed for computational reasons. In this case the first stage is typically designed for fast rejection of very abundant *outlier* objects, with a more complex second stage to discriminate between *target* classes.

Evaluating the recognition system involves analysing the classification accuracy, and the rate of *outlier* false acceptances. Importantly, a poor detector that does not detect a large fraction of *target* objects results in poor classification performance. In the opposite case, a very sensitive detector may pass an unacceptably large fraction of *outlier* objects to the classifier, which may for example result in high manual processing costs or computational overload.

The paper is structured as follows: Section 6.2.2 presents an analytic example to demonstrate how the two classifiers interact. A cost-based approach using ROC analysis demonstrates how system optimisation can be performed in evaluating the entire system. In Section 6.2.3 the multiple-class extension is discussed briefly, highlighting some problems that exist in extending the analysis to a large number of *target* classes. In Section 6.2.4, some experiments on real data are performed, consisting of a simple problem with 2 *target* classes, and a 4-class problem involving hand-written digit recognition. In Section 6.2.5 we briefly consider the case in which priors or costs cannot be defined precisely, discussing how different system configurations can be chosen in these situations. Conclusions are given in Section 6.2.6.

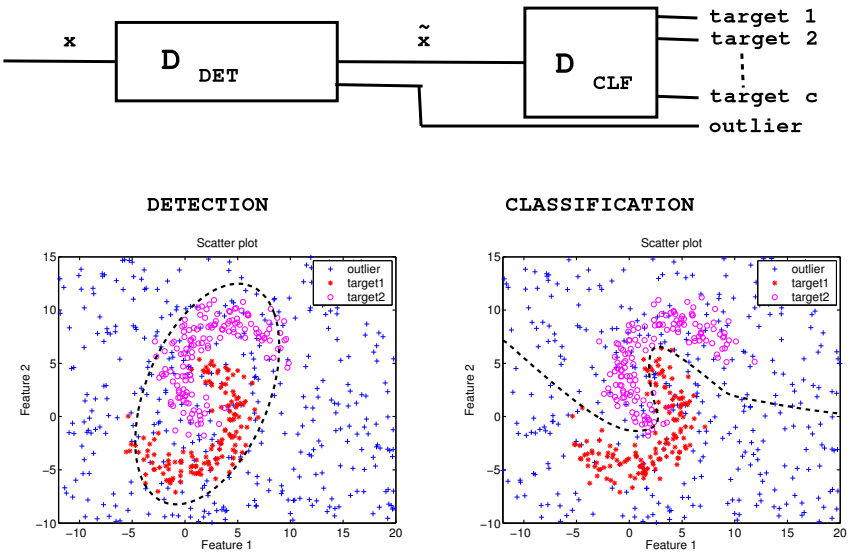


Figure 6.1: Illustrating a typical recognition system on a synthetic example. The scatter plots show a 2-dimensional synthetic example with two *target* classes, illustrating the detector in the left plot, and the classifier in the right.

6.2.2 The dependence between classifiers

Two-stage recognition systems

Consider a recognition task in which there are a number (n) *target* classes $\omega_{t1}, \omega_{t2}, \ldots, \omega_{tn}$, and an *outlier* class ω_o. A recognition system, as illustrated in Figure 6.1, has to classify these objects. A detector D_{DET} classifies incoming objects as either *target* (ω_t), or *outlier* via a detection threshold θ_d:

$$D_{DET}(\mathbf{x}) : \begin{cases} target & \text{if } f_{DET}(\mathbf{x}) > \theta_d \\ outlier & \text{otherwise} \end{cases} \tag{6.1}$$

The detector selects objects from $\mathbf{x}$ such that the input to D_{CLF} is $\tilde{\mathbf{x}}$.

$$\tilde{\mathbf{x}} = \{\mathbf{x} | f_{DET}(\mathbf{x}) > \theta_d\} \tag{6.2}$$

The classifier D_{CLF} then classifies incoming objects (according to $\tilde{\mathbf{x}}$) to any of the n *target* classes via the classification thresholds[3] $\theta_c^{t1}, \theta_c^{t2}, \ldots, \theta_c^{tn}$. The classifier outputs are weighted by classification thresholds. The classifier outputs $f_{CLF}(\tilde{\mathbf{x}})$ can then be written as:

$$[\theta_c^{t1} f_{CLF}(\omega_{t1}|\tilde{\mathbf{x}}), \theta_c^{t2}(\omega_{t2}|\tilde{\mathbf{x}}), \ldots, \theta_c^{tn} f_{CLF}(\omega_{tn}|\tilde{\mathbf{x}})] \tag{6.3}$$

Here $\sum_{i=1}^{n} \theta_c^{ti} = 1$. The final decision rule is then:

$$D_{CLF}(\tilde{\mathbf{x}}) = \text{argmax}_{i=1}^{n}\, \theta_c^{ti} f_{CLF}(\omega_{ti}|\tilde{\mathbf{x}}) \tag{6.4}$$

The primary distinction between this two-stage system and a multi-class single-stage recognition system is that the input to the classification stage in the two-stage case is a subset of the system input, whereas in the single-stage case all data is processed. We are considering the dependence (in terms of overall system performance) of the 2 stages, and how the system should be optimised.

One-dimensional example

In this section a simple 1-dimensional analytical example is studied in order to illustrate how the detection and classification stages are related. Two Gaussian-distributed *target* classes ω_{t1} and ω_{t2} are to be detected from a uniformly-distributed *outlier* class ω_o, and subsequently discriminated. The *target* classes have means of -1.50 and 1.50 respectively, and variances of 1.50. The ω_o class has a density of 0.05 across the domain x. The class conditional densities for ω_{t1}, ω_{t2} and ω_o are denoted $p(x|\omega_{t1})$, $p(x|\omega_{t2})$, and $p(x|\omega_o)$ respectively, with priors $p(\omega_{t1})$, $p(\omega_{t2})$, and $p(\omega_o)$, which are assumed equal here. For the total probability distribution of x therefore holds:

$$p(x) = p(\omega_{t1})p(x|\omega_{t1}) + p(\omega_{t2})p(x|\omega_{t2}) + p(\omega_o)p(x|\omega_o) \tag{6.5}$$

[3] In an n-class situation, the classification thresholds can be considered to be the weighting applied to the output posterior density estimates together with priors.

For this 1-dimensional data, the classifier is defined consisting of only a single threshold, denoted θ_c. The position of θ_c determines the classification performance, and can be used to set an operating point to achieve a specified false-negative rate FN_r (with respect to ω_{t1}) or false-positive rate (FP_r). These two errors are known as the Error of Type I and II respectively (ϵ_I and ϵ_{II}). As θ_c varies, so do the respective ϵ_I and ϵ_{II}, resulting in the ROC (receiver-operator curve [8]) between ω_{t1} and ω_{t2}. In a typical discrimination problem (ignoring the detector) across domain x, we can define ϵ_I and ϵ_{II} in terms of θ_c as:

$$\epsilon_I = 1 - \int_{-\infty}^{\infty} p(x|\omega_{t1}) I_1(x|\theta_c) dx, \; \epsilon_{II} = 1 - \int_{-\infty}^{\infty} p(x|\omega_{t2}) I_2(x|\theta_c) dx \tag{6.6}$$

The indicator functions $I_1(x|\theta)$ and $I_2(x|\theta)$ specify the relevant domain:

$$\begin{array}{l} I_1(\mathbf{x}|\theta_c) = 1 \text{ if } \; p(\omega_{t1})p(\mathbf{x}|\omega_{t1}) - p(\omega_{t2})p(\mathbf{x}|\omega_{t2}) < \theta_c, \; 0 \text{ otherwise} \\ I_2(\mathbf{x}|\theta_c) = 1 \text{ if } p(\omega_{t1})p(\mathbf{x}|\omega_{t1}) - p(\omega_{t2})p(\mathbf{x}|\omega_{t2}) \geq \theta_c, \; 0 \text{ otherwise} \end{array} \tag{6.7}$$

A two-stage recognition system consists of two sets of thresholds, namely a classification threshold θ_c (of which there are a number of thresholds according to the number of classes), and a detection threshold θ_d. Evaluating the recognition system involves estimating both classification performance (ϵ_I and ϵ_{II}), and the fraction of *outlier* objects incorrectly classified as *target*, denoted FP_r^o. Thus one axis of the evaluation is concerned with how well the system performs at detecting and discriminating *target* classes, and the other is concerned with the amount of false alarms that the system must deal with. Therefore the system must be evaluated with respect to ϵ_I, ϵ_{II}, and FP_r^o. In this simple example, we can write these as:

$$\begin{array}{rcl} \epsilon_I & = & 1 - \int_{-\infty}^{\infty} p(x|\omega_{t1}) I_1(x|\theta_c) I_R(x|\theta_d, \omega_{t1}) dx \\ \epsilon_{II} & = & 1 - \int_{-\infty}^{\infty} p(x|\omega_{t2}) I_2(x|\theta_c) I_R(x|\theta_d, \omega_{t2}) dx \\ FP_r^o & = & \int_{-\infty}^{\infty} p(x|\omega_o) I_1(x|\theta_c) I_R(x|\theta_d, \omega_{t1}) + p(x|\omega_o) I_2(x|\theta_c) I_R(x|\theta_d, \omega_{t2}) dx \end{array} \tag{6.8}$$

$$I_R(\mathbf{x}|\theta_d, \omega) = 1 \; if \; p(\mathbf{x}|\omega) > \theta_d, \; 0 \; otherwise \tag{6.9}$$

Equation 6.8 yields the full operating characteristics of the system, shown in Figures 6.2 and 6.3 for the example. Referring first to Figure 6.2, this shows how the system operating characteristics vary for a number of fixed detection thresholds. The top row illustrates the position of the detection threshold, and the bottom row shows ϵ_I, ϵ_{II}, and FP_r^o for all classification thresholds (similar to standard ROC analysis, except an additional dimension is introduced to account for the detection threshold). In these plots, it is desirable for ϵ_I, ϵ_{II}, and FP_r^o to be minimal, indicating good classification and detection.

In Figure 6.2, as θ_d is increased, the plots show how FP_r^o progressively decreases. In the left-column, a very sensitive detector is used, with θ_d placed in the tails of the *target* distribution. It is clear that the classification performance is almost maximal for this threshold, but FP_r^o is very high i.e. the system will

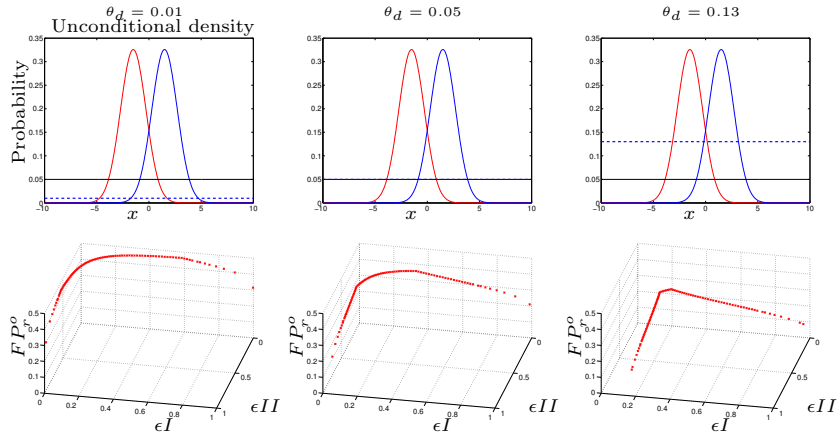


Figure 6.2: Operating characteristics for a fixed θ_d, and varying θ_c. The left column is where $\theta_d = 0.01$, followed by $\theta_d = 0.05$ in the middle column, and $\theta_d = 0.13$ in the right column. The top row plots illustrate the distribution, with two Gaussian *target* classes, and a uniformly distributed *outlier* class. The position of the detection threshold is shown via the dotted line. The full operating characteristics for all possible θ_c are shown in the bottom row.

accept a very high percentage of *outlier* objects. The centre column plots show the case for which a higher detection threshold has been used ($\theta_d = 0.05$), resulting in a substantially lower FP_r^o, for a small sacrifice in classification performance. The third column shows a situation in which θ_d is again increased, resulting in a further decrease in classification performance. In this case the detector only accepts very probable *target* objects, reducing the volume of the *target* class decision space, at the expense of all *target* objects appearing outside the decision boundary. The left plot of Figure 6.3 shows the operating characteristics for all combinations of θ_c and θ_d. Next we show how using the full operating characteristic can be advantageous in system design.

Cost-based analysis

From the system perspective, the cost of misclassifying a ω_t object (as *outlier*) is c_t, and the cost of misclassifying a ω_o object (as *target*) is c_o. The individual *target* class misclassification costs can be written as $c_{t1}, c_{t2}, \ldots c_{t_n}$, which must sum to c_t together with the priors (note that we do not consider the entire loss matrix as defined in [2], but only consider the loss incurred due to misclassification, irrespective of the class to which it is assigned). The expected overall system loss L can be written as:

$$L = c_t p(\omega_t) FN_r + c_o p(\omega_o) FP_r^o, \; = \sum_{i=1}^{n} c_{t_i} p(\omega_{t_i}) FN_r^{t_i} + c_o p(\omega_o) FP_r^o, \;\; \sum_{i=1}^{n} c_{t_i} = c_t \tag{6.10}$$

The priors are denoted $p(\omega_t)$ and $p(\omega_o)$, and the false negative rate of ω_t is denoted FN_r. The *target* class misclassification costs are denoted c_{t_i} for *target*

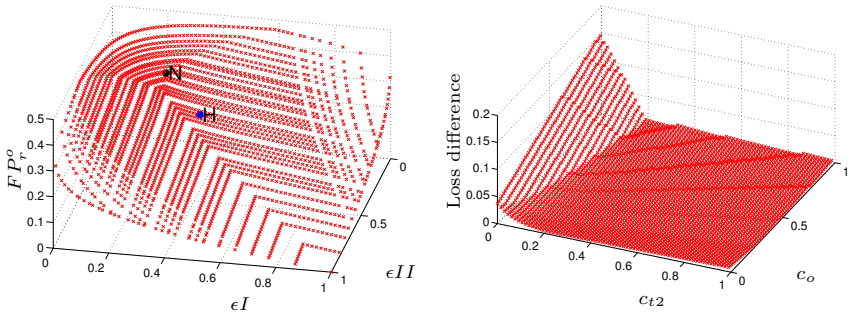


Figure 6.3: Results of analytic experiment. The left plot shows the full operating characteristics, with ϵ_I plotted against ϵ_{II}, and FP_r^o. The right plot shows the loss difference between an independent and holistic design approach for all combinations of c_{t2}, and c_o over a $\{0, 1\}$ range, where c_{t1} is fixed to 0.55.

class ω_{t_i}. Cost-based classifier design involves minimising of L for the given costs, resulting in the optimal threshold values. The ROC is a tool that can be used to facilitate this minimisation, since it consists of performances for all possible threshold values (all FN_r and FP_r results). In a 2-class problem, the costs (and priors) specify the gradient of the cost line (also known as an *iso-performance* line as defined in [10]), and the intersection of the normal of this line with the ROC (plotting FN_r against FP_r) results in the optimal operating point[4].

We now demonstrate a cost-analysis for the example in order to emphasise the importance of designing the entire system holistically. Two different design approaches are compared, the first of which we refer to as the *independent* approach, and the second as the *holistic* approach. In the first case, we optimise the recognition and classification stages independently, and compare the expected system loss to the second case, in which the entire system is optimised holistically. We assume that the cost specification for the recognition system is such that misclassifying a ω_t object has a cost of 5, and the cost of classifying a ω_o object as *target* is 10. Among the two *target* classes ω_{t1} and ω_{t2}, these have misclassification costs of 2 and 3 respectively (summing to 5), i.e. ω_{t2} is favoured. From Equation 6.10, we can write the system loss (assuming equal priors) for the chosen θ_c and θ_d as $L(\theta_c, \theta_d) = 2\epsilon_I(\theta_c, \theta_d) + 3\epsilon_{II}(\theta_c, \theta_d) + 10FP_r^o(\theta_c, \theta_d)$. In the *independent* approach, the detector is optimised using ω_t and ω_o data only (with operating characteristics generated for these classes only). The classifier is then optimised on ω_{t1} and ω_{t2}. The corresponding thresholds are indicated by the point marked **N** in the left plot of Figure 6.3. In the holistic approach, ω_{t1}, ω_{t2},

[4]We deal with multi-dimensional ROC plots in this paper. Cost-based optimisation involves intersecting a plane (the gradient based on the cost associated with misclassifying each class) with the multi-dimensional ROC surface, resulting in optimised thresholds.

and ω_o are analysed simultaneously in the optimisation, resulting in the point marked **H**. The two points **N** and **H** are significantly apart on the operating characteristic. In the *independent* approach, the overall expected loss is thus **4.18**, and in the *holistic* approach, the loss is **4.02**. Thus independent approach is sub-optimal here. Depending on the problem and the costs, the *independent* approach may vary in the degree of sub-optimality. To assess how the holistic approach will improve performance in general, refer to the right plot of Figure 6.3. This plot shows the difference between the *independent* and *holistic* loss performances (where a positive score indicates superiority of the *holistic* approach) for all combinations of costs over a range. The cost c_{t1} is fixed to 0.55, and c_{t2} and c_o are varied for all combinations over the $\{0, 1\}$ range. It can be seen that for this artificial example, only imbalanced costs result in significant improvements. In the experiments, it will be shown models that do not fit the data well in real problems can benefit even more from this approach, including balanced cases.

6.2.3 Multiple class extension

The analytic example involved a recognition system with 2 *target* classes, resulting in a 3-dimensional ROC surface. As the number of *target* classes increase, the dimensionality of the ROC increases. The analysis extends to any number of classes [11]. However, as the number of dimensions increase, the computational burden becomes infeasible [5]. In this paper, experiments involved up to 3 *target* classes. In this case, the processing costs were already very high. and only a very sparsely sampled ROC could be generated. Extending this analysis to N classes would be infeasible. This is the topic of future work, exploring approaches that can be used to either approximate the full ROC, or to use search techniques in optimising the thresholds. Attention is drawn to [6], in which an initial set of thresholds is used, and a hill-climbing greedy-search is used.

6.2.4 Experiments

In this section a number of experiments are conducted on real data in order to demonstrate the holistic system design approach practically, and how model (or system configuration) selection can be performed. Two datasets are used, described as follows:

- *Banana*: A simple 2 dimensional problem with 2 *target* classes distributed non-linearly (the banana distribution [4]), in which there are 600 examples each of ω_{t1} and ω_{t2}, and 2400 *outlier* examples. The distribution is shown in Figure 6.1.

- *Mfeat*: This is a dataset consisting of examples of ten handwritten digits, originating from Dutch utility maps[5]. In this dataset, Fourier components have been extracted from the original images, resulting in a 76-dimensional representation of each digit. 200 examples of each digit are

[5]Available at ftp://ftp.ics.uci.edu/pub/machine-learning-databases/mfeat/

available. In these experiments, digits *3*, *4* and *8* are to be distinguished (i.e. 3 *target* classes ω_{t1}, ω_{t2}, and ω_{t3}), distributed among all other digit classes, which are considered to be *outlier*.

We follow the same analysis approach as in Section 6.2.2. Classification and detection thresholds are generated across the full range. In the *Banana* case, 200 evenly sampled classification thresholds are used, and similarly 100 detection thresholds are used. For computational reasons, the *Mfeat* experiments only uses 10 detection thresholds, and 12 samples per classification threshold. Each experiment involves a 10-fold randomised hold-out procedure, with 80% of the data used in training, and the remainder for testing. The evaluation consists of evaluating the loss incurred for a number of chosen misclassification costs, using the ROC to find an optimal set of thresholds. In this evaluation it is assumed that the costs (and priors) are known beforehand, and as in Section 6.2.2, we only consider misclassification costs, applying Equation 6.10.

In the *Banana* experiments, 3 different system configurations are implemented, comparing the *independent* and *holistic* approaches for each case. The same detector is used for all 3 configurations, consisting of a Gaussian one class classifier (OCC) [12]. Three different classifier models are used, consisting of a Bayes linear, quadratic, and mixture of Gaussians classifier (with two mixtures per class), denoted LDC, QDC, and MOG respectively. In Table 6.1 the *Banana* experimental results are shown for 4 different system costs. These are shown in the four right-most columns, with the costs denoted $[c_{t1}, c_{t2}, c_o]$. For all 3 system configurations, the *holistic* design approach results in a lower overall expected loss than the *independent* approach. In some cases the difference in performance is not significant (see the MOG results for the case in which $c_{t1} = 3.0$, $c_{t2} = 1.0$, and $c_o = 4.0$). These experiments show that the benefit of an overall design approach can in many cases result in significant improvements in performance.

A similar set of experiments are conducted for the *Mfeat* problem, with costs denoted $[c_{t1}, c_{t2}, c_{t3}, c_o]$. Results are shown for four different cost specifications in the right-most columns of Table 6.1. Three different system configurations are considered, and in each case the *independent* and *holistic* design approaches are compared. The first configuration consists of a principal component analysis (PCA) mapping with 3 components and a Gaussian OCC as the detector, followed by a Fisher mapping and LDC as the classifier. The second configuration uses a 3-component PCA mapping Gaussian OCC for the detector, and a 3-component PCA LDC for the classifier. Finally the third system consists of a 5-component PCA with Gaussian OCC detector, and a 2-component PCA MOG classifier with 2 mixtures for the classifier. As before, the *holistic* approach consistently results in either a similar or lower overall loss compared to the *independent* approach. Once again, the improvement is dependent on the cost specification. For costs $[1, 8, 1, 10]$ (favouring ω_{t2}) and $[1, 1, 1, 12]$ (favouring ω_o), there is no significant improvement in using the *holistic* approach for all

Detector	**Classifier**	Cost 1	Cost 2	Cost 3	Cost 4
Banana		$[5, 5, 10]$	$[3, 1, 4]$	$[1, 3, 4]$	$[1, 1, 4]$
Gauss	LDC I	0.081 ± 0.009	0.370 ± 0.049	0.233 ± 0.056	0.244 ± 0.046
Gauss	LDC H	0.067 ± 0.008	0.326 ± 0.039	0.171 ± 0.015	0.189 ± 0.027
Gauss	QDC I	0.089 ± 0.017	0.418 ± 0.051	0.260 ± 0.060	0.265 ± 0.053
Gauss	QDC H	0.072 ± 0.010	0.354 ± 0.036	0.179 ± 0.025	0.182 ± 0.030
Gauss	MOG I	0.059 ± 0.008	0.252 ± 0.033	0.206 ± 0.032	0.205 ± 0.030
Gauss	MOG H	0.049 ± 0.007	0.230 ± 0.035	0.170 ± 0.019	0.169 ± 0.021
Mfeat		$[1, 1, 1, 3]$	$[8, 1, 1, 10]$	$[1, 8, 1, 10]$	$[1, 1, 1, 12]$
PCA3 Gauss	Fisher LDC I	0.648 ± 0.050	0.212 ± 0.018	0.225 ± 0.017	1.385 ± 0.316
PCA3 Gauss	Fisher LDC H	0.547 ± 0.110	0.146 ± 0.014	0.223 ± 0.017	1.317 ± 0.435
PCA3 Gauss	PCA3 LDC I	0.654 ± 0.053	0.214 ± 0.018	0.225 ± 0.017	1.389 ± 0.316
PCA3 Gauss	PCA3 LDC H	0.551 ± 0.110	0.146 ± 0.015	0.224 ± 0.017	1.305 ± 0.432
PCA5 Gauss	PCA2 MOG2 I	0.442 ± 0.029	0.146 ± 0.011	0.154 ± 0.011	0.929 ± 0.202
PCA5 Gauss	PCA2 MOG2 H	0.380 ± 0.079	0.112 ± 0.024	0.148 ± 0.018	0.847 ± 0.124

Table 6.1: Results of cost-based analysis for the *Banana* and *Mfeat* datasets, comparing an *independent* (I) and *holistic* (H) design approach for a number of different system configurations (low scores are favourable). Standard deviations are shown.

3 systems. However, when the costs are in favour of ω_{t1}, the holistic approach leads to a significantly lower system loss. This suggests that the ω_{t1} threshold has more effect over the detection performance. In this case θ_d should be adjusted accordingly for optimal performance. The same observation is made for balanced costs $[1, 1, 1, 3]$. An interesting observation made in these experiments is models that do not fit the data well (e.g. the LDC in the *Banana* experiments, compared to MOG), tend to benefit more from the holistic optimisation, suggesting that the interaction is more prominent for all costs.

6.2.5 Imprecise environments

The approach taken thus far showed that, given both misclassification costs and priors, the optimal set of thresholds can be found. In many practical situations the costs or priors cannot be obtained or specified precisely [10]. In these situations we may still wish to choose the best system configuration, and have some idea of a good set of system thresholds that may, for example, be suitable for a range of operating conditions or costs (see [1] and [3]). We do not go into more detail here due to space constraints, but emphasise the fact that real problems are often within an imprecise setting, requiring an alternative evaluation to the cost-based approach. One strategy for this situation is to compute the AUC (Area Under the ROC curve) for a range operating points. An integrated error results that is useful for model selection. The next step is to choose thresholds, which may for example be specified by considering operating regions that are relatively insensitive to changes in cost or priors.

6.2.6 Conclusion

A two-stage recognition system was considered as an MCS, consisting of a detection and classification stage, with the objective of optimising the overall system. An analysis of a simple analytic problem was performed, in which the full operating characteristics were computed for all combinations of detection

and classification thresholds. The holistic design approach was compared to the case in which the two stages are designed independently, showing that the holistic approach may result in a lower expected loss. The N-class extension was discussed, highlighting the computational difficulties in scaling the analysis to any number of classes. Some experiments with real data were then undertaken for a number of system configurations to demonstrate practical application of the analysis, consistently demonstrating the advantage of the holistic design approach. It was observed that the performance improvements vary according to the cost specification, and the respective degree of interference a class may impose on the detection stage. Models that fit the data well only seem to benefit for imbalanced costs/priors, whereas ill-fitting models can result in improvements for any costs. Finally, a short discussion on application of the methodology to imprecise environments was given. Future work includes exploring efficient multi-class ROC analysis, and application to an imprecise environment.

Acknowledgements: This research is/was supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs.

Bibliography

[1] N.M. Adams and D.J. Hand. Comparing classifiers when misallocation costs are uncertain. *Pattern Recognition*, 32(7):1139–1147, 1999.

[2] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press Inc., New York, first edition, 1995.

[3] A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[4] R.P.W. Duin, P. Juszcak, P. Paclik, E. Pekalska, D. de Ridder, and D.M.J. Tax. Prtools, a matlab toolbox for pattern recognition, January 2004.

[5] C. Ferri, J. Hernandez-Orallo, and M.A. Salido. Volume under the roc surface for multi-class problems. *Proc. of 14th European Conference on Machine Learning*, pages 108–120, 2003.

[6] N. Lachiche and P. Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. *Proc. 20th Int. Conf. on Machine Learning, Washington DC*, pages 416–423, 2003.

[7] A. Lipnickas, J.S. da Costa, and C.D. Bocaniala. FDI based on two stage classifiers for fault diagnosis of valve actuators. *11th Int. Power Electronics and Motion Control Conference*, pages 3,147–153, September 2004.

[8] C. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, 3(4), 1978.

[9] P. Paclík. Building road sign classifiers. *PhD thesis, CTU Prague, Czech Republic*, December 2004.

[10] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42:203–231, 2001.

[11] A. Srinivasan. Note on the location of optimal classifiers in N-dimensional ROC space. *Oxford University Computing Laboratory Technical report PRG-TR-2-99*, October 1999.

[12] D.M.J. Tax. One-class classification. *PhD thesis, Delft Technical University, The Netherlands*, June 2001.

Summary

Operating characteristics for the design and optimisation of classification systems

This thesis has consisted of a number of published conference and journal papers in the area of classifier operating characteristics. As a whole, the thesis shows how powerful and versatile operating characteristics are at aiding in the design of classification systems. New areas have also been tackled, with strategies and algorithms proposed that broaden the scope of pattern recognition to new problems. A conclusion that can be made from this body of work is that it is not only important to focus on a good design of the classifier model, but it can also be very beneficial to evaluate and optimise the system to suit operating conditions and circumstances. The strategies and algorithms proposed may form the basis for further research in this area.

In the second chapter, well-known 2-class operating characteristics were discussed, popularly known as Receiver Operator Characteristics (ROC). The first part considered the use of the popular Area Under the ROC (AUC) measure in the case that prior probabilities vary (as opposed to being fixed). The research showed that even though it is important to consider the integrated performance over the expected range of conditions, it is also important to consider the sensitivity to the changes in conditions. Experiments demonstrated that in some cases, classifiers that are competitive in terms of AUC, have significantly different sensitivities. The second part of the chapter considered extending ROC analysis to problems involving evaluation using *precision-recall* analysis. This is commonly used in applications such as retrieval and rare event detection, where performance measurement using standard error-rate does not suffice. The research showed that Precision-Recall Operating Characteristics (P-ROC) form a 3-dimensional surface, between the Precision, Recall, and degree of class skewing between class abundances. An analysis was presented that allows these problems to be evaluated in imprecise conditions, resulting in performance measures similar in concept to the AUC.

The multiclass generalisation of 2-class operating characteristics has thus far received very little attention, even though it has wide application. The third chapter investigated some theoretical extensions of popular 2-class operating characteristics to the multi-class case. The first contribution in this area was a

generalised Neyman-Pearson classifier optimisation approach. This optimisation is frequently used in the 2-class case where the classification error is fixed for one class, and minimised for the other. An algorithm was proposed that can be applied to the multiclass case subsequent to the computation of the multiclass ROC. The approach allows one or more classifier outputs to be specified, followed by minimisation of the remaining respective classification errors. However, a solution is not guaranteed if more that one output is specified, because it may not exist. The second part of this chapter considered the extension of the AUC measure to the multiclass case. A simplified approach was proposed that considers overall inter-class performance only (ignoring intra-class errors). For this study, it was important to consider the bounds of the measurement between a random and perfect classifier, since they vary as a function of the number of classes. These were calculated, and a numerical integration approach used to measure the volume under the ROC hypersurface. The approach can be applied to any trained classifier. Even though the bounds for the full generalised case have been calculated, a practical methodology for computing the volumes is still an open area.

The fourth chapter of the thesis dealt with the fact that generalising ROC analysis to large numbers of classes is computationally restrictive. In fact, it was shown that the computational demands grow exponentially with the number of classes. A number of algorithms were presented that perform cost-sensitive optimisation using efficient search approaches, circumventing the necessity to construct the ROC. In these cost-sensitive problems, the objective is to choose a new operating point for the classifier to suit a new cost/risk scenario. The search approaches are, however, susceptible to local minima. An approach was then proposed that involves constructing 2-class ROC's between all class pairs. The ROC pairs that result in the most optimal performance for the scenario are selected, implicitly considering the most important pairs of interactions (between classes). The algorithm was found to work well, but is sub-optimal if there are many interactions, which is compensated for to an extent by using a post-processing search step. The second part of this chapter presented one of the primary thesis contributions. This involves a classifier perturbation-analysis approach that can be used to efficiently approximate the full multiclass operating characteristic using a decomposition approach. The perturbation analysis considers the classifier performance over many different operating points in order to discover which classifier outputs tend to interact strongly, and which are more independent. A considerable amount of experimentation with multiclass problems was carried out, considering performance at different operating points. This revealed that many ROC dimensions (each confusion matrix output becomes an ROC dimension) are often (approximately) independent, or interact in independent groups. The consequence of this is that the ROC can be decomposed into a number of lower-dimensional groups of dimensions that are far more efficient to compute (essentially computing a number of sub-ROC's). An algorithm was presented that efficiently discovers the decomposition, and a number of studies investigated the impact of ignoring smaller

interactions, which is important to obtain approximate decompositions to limit computational complexity. Experiments demonstrated just how powerful this approach can be at simplifying some problems down to their intrinsic complexities. This type of approach could form the basis for much further multiclass ROC research.

In the fifth chapter, the topic of ill-defined conditions was studied in the context of classification systems. The first part proposed a 2-stage approach to designing classifier systems that can both discriminate between known classes, and reject objects from new/unseen classes. This methodology is an alternative to the standard approach that uses a single model to perform both tasks. It was found that optimising different models for the tasks of classification and rejection respectively can be beneficial. The second part of the chapter consisted of a study into the operating characteristics that exist as a consequence of having both classification and reject thresholds. This is an important topic because increasing the rejection performance decreases classification performance (and vice-versa). Operating characteristics can be used to select the most appropriate set of thresholds for a given system, accounting for the inherent interaction.

In the sixth chapter, the thesis showed that operating characteristics can also be applied to multi-stage classification systems, modeling both the interaction between classes and stages. In particular, a two-stage recognition system was studied, consisting of a detection and classification stage. A multi-dimensional operating characteristic showed that this holistic approach to optimising the entire system yields superior performance to a more traditional independent approach. Generalising this methodology to other multistage systems remains an open challenge.

This thesis has thus considered and contributed to a number of areas in pattern recognition pertaining to operating characteristics. It is clear that research into operating characteristics is gaining momentum, and will become one of the most useful tools for evaluating and optimising classification systems as pattern recognition extends to new exciting areas.

Summary of the thesis: *"Operating characteristics for the design and optimisation of classification systems"*.

T.C.W. Landgrebe, Delft, October 2007.

Samenvatting

Operating karakterisieken voor het ontwerp en de optimalisatie van classificatie systemen

Dit proefschrift omvat een aantal conferentie- en tijdschrift-publicaties op het gebied van classificator keurings (operating) karakteristieken. Het proefschift laat zien hoe krachtig en veelzijdig operating karakteristieken zijn bij het ontwerp van classificatie systemen. Ook nieuwe onderwerpen zijn aangepakt, strategieen en algoritmes zijn voorgesteld die het toepassingsgebied van de patroonherkenning verder uitbreiden. Eén conclusie die uit dit werk getrokken kan worden is dat het niet alleen belangrijk is om zich te concentreren op het ontwerp van de classificator, maar dat het ook erg nuttig kan zijn om het systeem te optimaliseren voor de gebruikssituaties en -omstandigheden. De voorgestelde strategieën en algoritmes kunnen de basis vormen voor verder onderzoek in dit gebied.

In het tweede hoofdstuk worden bekende twee-klasse karakteristieken besproken, die algemeen bekend zijn als Receiver Operating Characteristics (ROC), of Operating karakeristieken. Het eerste deel beschouwt het gebruik van het populaire 'Area Under the ROC' maat voor het geval dat de a priori kansen varieren (in tegenstelling tot vastgelegde prior kansen). Het onderzoek laat zien dat, hoewel het belangrijk is om de prestaties over het hele verwachte toepassings condities te integreren, het erg belangrijk is om de gevoeligheid voor veranderingen in die condities te bekijken. Experimenten laten zien dat in sommige gevallen de klassificatoren die vergelijkbare AUC hebben, geheel verschillende gevoeligheden bezitten. Het tweede gedeelte van het hoofdstuk bekijkt de uitbreiding van de ROC analyse naar problemen die met behulp van *precision-recall* worden geanalyseerd. Dit wordt veel gebruikt in terugzoeksystemen of in de detectie van zeldzame gebeurtenissen, waar de standaard fout schatting met behulp van de klassificatie fout niet voldoet. Het onderzoek laat zien dat de Precision-Recall operating karakteristieken (P-ROC) een driedimensionale oppervlak definieert opgespannen door de Precision, Recall en de mate van onevenwichtigheid in de klassengroottes. Er wordt een analyse getoond waarin een evaluatie mogelijk is voor situaties met onbekende klassegroottes. Dit resulteert in een maat die op de AUC lijkt.

De generalisatie van de twee-klasse karakeristieken naar meerdere klassen

heeft tot op heden weinig aandacht gekregen, hoewel het erg relevant is in praktijk. Het derde hoofdstuk onderzoekt enkele theoretische uitbreidingen van populaire twee-klasse karakteristieken naar meerdere klassen. De eerste bijdrage is een gegeneraliseerde Neyman-Pearson klassificatie optimalisatie. Deze optimalisatie wordt vaak in het twee-klasse probleem toegepast waarbij de klassificatie fout van één klasse gefixeerd wordt en de klassificatie fout van de andere klasse geminimaliseerd wordt. Een algoritme is voorgesteld dat kan worden toegepast op het multi-klasse geval na de berekening van de multi-klasse ROC. Deze aanpak maakt het mogelijk om één of meerdere klasse-fouten te specificeren en de overigen te minimaliseren. Een oplossing is niet gegarandeerd wanneer er meer dan één fout gespecificeerd is, omdat het niet hoeft te bestaan. Het tweede gedeelte van dit hoofdstuk bekijkt de uitbreiding van de AUC maat naar het meer-klasse probleem. Een versimpelde aanpak is voorgesteld dat alleen de totale fouten tussen twee klassen beschouwd (en daarbij fouten binnen klassen negeert). Hiervoor is het belangrijk de limieten op het verschil tussen een willekeurige en een perfecte classificator te beschouwen, en een numerieke integratie methode is gebruikt om het volume onder de multi-dimensionele ROC curve te berekenen. Dit kan op elk mogelijke classificator worden toegepast. Hoewel de limieten voor het volledige algemene geval zijn berekend, is een praktische methodology om de volumes te berekenen nog steeds een punt van onderzoek.

Het vierde hoofdstuk van het proefschrift gaat over het feit dat de gegeneraliseerde ROC analyse computationeel zeer duur is voor grote hoeveelheden klassen. Er wordt aangetoond dat de computationele eisen exponentieel toenemen met het aantal klassen. Een aantal algoritmes wordt gepresenteerd die kost-gevoelige optimalisaties doen met behulp van efficiente zoek algorithmes, waarbij de dure constructie van de ROC wordt vermeden. In deze kost-gevoelige problemen is het doel om een nieuw operating point voor de classificator te kiezen dat past bij de nieuwe kosten. De methodes zijn helaas gevoelig voor locale optima. Eén van de voorgestelde methodes maakt gebruik van twee-klasse ROC tussen alle paren van klassen. De set van ROC paren die de kleinste fout geven voor het gegeven scenario wordt geselecteerd, waarbij impliciet de belangrijkste interactie paren (tussen de klassen) wordt meegenomen. Het algoritme werkt goed, maar is suboptimaal wanneer er veel interacties zijn, en dit kan gedeeltelijk gecompenseerd worden met behulp van een post-processing zoek stap. Het tweede gedeelte van het hoofdstuk presenteert een van de belangrijkste bijdragen van dit proefschrift. Het bevat een classificator verstorings-analyse benadering die gebruikt kan worden om de volledige multi-klasse operating characteristic efficient te kunnen schatten met behulp van een decompositie benadering. De verstorings-analyse beschouwd de classificators prestaties op veel verschillende operating points om uit te vinden welke classificator uitvoeren sterk met elkaar interageren, en welke meer onafhankelijk zijn. Een substantieel aantal experimenten met multi-klasse problemen was gedaan, waarbij de prestaties bij verschillende operating points gemeten is. De experimenten tonen aan dat veel ROC dimensies (elke element uit de verwar-

ringsmatrix kan als een ROC dimensie worden gezien) vaak (bij benadering) onafhankelijk zijn, of in onafhankelijke groepen uiteenvallen. De consequentie daarvan is dat de ROC kan worden ontbonden in een aantal laag-dimensionale groepen die veel efficienter zijn om te berekenen (in wezen het berekenen van een aantal sub-ROC's). Er is een efficient algoritme gepresenteerd dat een decompositie levert, en een aantal studies laat de invloed van het negeren van de kleinere interacties zien, wat belangrijk is voor het vinden van benaderingen van de decompositie om de computationele complexiteit te beperken. Experimenten tonen aan hoe krachtig deze aanpak kan zijn om sommige problemen te simplificeren.

In het vijfde hoofdstuk wordt het onderwerp van slecht-gedefinieerde condities bestudeerd in de context van classificatie systemen. Het eerste gedeelte stelt een twee-stappen procedure voor voor het ontwerp van systemen die zowel tussen bekende klassen kunnen discrimineren als objecten uit nieuwe/onbekende klassen kunnen verwerpen. Deze methodologie is een alternatief voor de standaard aanpak waarin een enkel model wordt gebruikt voor beide taken. Het blijkt dat het erg voordelig kan zijn om verschillende modellen te optimaliseren voor het classificeren en het verwerpen. Het tweede gedeelte van het hoofdstuk bevat een studie van de operating karakteristieken die ontstaan wanneer classificatie en verwerping worden gecombineerd. Dit is een belangrijk onderwerp omdat bij het vergroten van de verwerpings fractie de klassificatie prestaties verminderen (en vise versa). De operating karakteristieken kunnen gebruikt worden om de meest toepasselijke drempelwaardes te kiezen voor het systeem, waarbij de inherente klasse-interacties worden meegenomen.

In het zesde hoofdstuk wordt getoond dat operating karakteristieken ook toegepast kunnen worden op meerstaps classificatie systemen, waarbij zowel de interactie tussen de klassen als de verschillende statia gemodelleerd wordt. In het bijzonder is een twee-klasse classificatie systeem bestudeerd, dat uit een detectie en een classificatie stap bestaat. Een multi-dimensionaal operating karacteristiek laat zien dat deze holistische aanpak om het gehele systeem te optimaliseren betere prestaties oplevert dan de traditionelere aanpak. Het is nog een uitdaging deze aanpak te generaliseren naar andere meerstaps systemen.

Dit proefschrift heeft het onderwerp van operating karakteristieken in de patroonherkenning onderzocht en uitgebreid. Het is duidelijk dat het onderzoek op dit gebied momentum wint, en dat het een van de nuttigste gereedschappen zal worden voor het evalueren en optimaliseren van classificatie systemen wanneer de patroonherkenning zich meer uitbreid naar nieuwe en interessante gebieden.

Samenvatting van het proefschrift: *"Operating characteristics for the design and optimisation of classification systems"*.

T.C.W. Landgrebe, Delft, Oktober 2007.

Curriculum Vitae

Thomas Christopher Wolfgang Landgrebe was born in Roodepoort, South Africa, on 24 July 1977. In 1995 he studied electrical engineering at the University of the Witwatersrand, Johannesburg, obtaining a BSc (Eng) degree. An MSc (Eng) degree was then obtained at the same University in 2002, in the area of stereo computer vision. Over this period, several industrial projects were performed, including electronic hardware design, software engineering, control systems design, and signal/image processing. From 2001 to 2003 he worked for the research and development division of De Beers Group Services, DebTech, in the area of video-based surveillance for security. Topics included stereo-object tracking, multiple-object tracking, segmentation, enhancement, and video-based software development.

In 2003 he started his Ph.D. research with the Pattern Recognition group at the Information and Communication Theory at Delft Technical University. The research was funded by De Beers, involving spectral pattern recognition for the discrimination of minerals. The Ph.D. research focus was in the area of operating characteristics and ill-defined problems, supervised by Prof. Robert Duin.

In 2005 he returned to De Beers to join a multi-disciplinary team to develop a sorting machine based on hyper-spectroscopy for mineral sorting. His primary role was pattern recognition and image processing. In 2007 he became project manager of a multi-camera particle-analysis system for size and shape analysis.

Acknowledgments

Put acknowledgements here.