Chapter 7

DECISION SUPPORT FOR DATA MINING

An introduction to ROC analysis and its applications

Peter Flach, Hendrik Blockeel, Cèsar Ferri, José Hernández-Orallo, and Jan Struyf

Abstract: In this chapter we give an introduction to ROC ('receiver operating characteristics') analysis and its applications to data mining. We argue that ROC analysis provides decision support for data mining in several ways. For model selection, ROC analysis establishes a method to determine the optimal model once the operating characteristics for the model deployment context are known. We also show how ROC analysis can aid in constructing and refining models in the modeling stage.

1. INTRODUCTION

Consider the following, fairly standard scenario for solving a particular data mining problem. The end-user provides the data miner with training data; the data miner builds a number of models from the training data; the data miner evaluates the models by means of cross-validation or a hold-out test set; finally, the best model is selected and presented to the end-user to provide predictions on new data. In this scenario we can distinguish the phases of *model construction* and *model evaluation and selection*. A variant of this scenario occurs when we allow approaches such as bagging, which effectively build new models by *model combination*.

In this chapter we consider a different scenario: the end-user is presented not with a single model, but with a collection of models *together with their operating characteristics*. These operating characteristics describe, for instance, whether a model is more accurate on the positives or more accurate on the negatives. Once the operating characteristics of the deployment context are known, i.e., the class distribution and the misclassification costs associated with each class, the end-user decides which model in the collection is optimal for that deployment context. The decision as to which model is best is thus taken out of the data mining stage and delayed until the deployment stage. Support for making this decision is provided by ROC analysis, where ROC stands for 'receiver operating characteristics'.

From this point of view, ROC analysis is a decision support technique. However, it is possible to utilize it while building models; more precisely, to have the data mining process build models with the specific aim of obtaining good ROC analysis performance. Thus, the decision support technique is integrated into the data mining process, instead of being used merely as a post-processing technique. This can significantly improve model performance, and as such it is a concrete illustration of the 'decision support for data mining' approach mentioned in Chapter 4, and of the claim made there that such integration can lead to improved overall performance.

The chapter is structured as follows. In Section 2 we give a brief introduction to ROC analysis. Section 3 describes how ROC analysis provides support for model selection. In Section 4 we describe some novel applications of ROC analysis in the areas of model construction, combination and refinement. Section 5 concludes the chapter.

2. WHAT IS ROC ANALYSIS?

Consider a set of examples labeled positive and negative, and a classifier predicting the label for each example (the choice of which class is called positive is arbitrary from the point of view of ROC analysis). A positive (negative) example that is correctly classified by the classifier is called a true positive (true negative); a positive (negative) example that is incorrectly classified is called a false negative (false positive). These numbers can be organized in a contingency table (*Table 7-1*). If we fix the number of examples (or if we replace absolute frequencies – counts – with relative frequencies) such a table has three degrees of freedom; that is, three out of four numbers can be freely chosen. Notice that the descending diagonal (left-to-right) in the table represents correct predictions, while the ascending diagonal represents incorrect predictions. Clearly, the best situation we can get is to have only 0's on the ascending diagonal.

Table 7-1. A contingency table or confusion matrix.

	Predicted positives (*PPos*)	Predicted negatives (*PNeg*)
Actual positives (*Pos*)	True positives (*TP*)	False negatives (*FN*)
Actual negatives (*Neg*)	False positives (*FP*)	True negatives (*TN*)

From the contingency table we can calculate other metrics, which can be used in various ways to evaluate the performance of the classifier on the dataset. The true positive rate (true negative rate) is the proportion of positives (negatives) correctly classified. Similarly, the false positive rate (false negative rate) is the proportion of negatives (positives) incorrectly classified. So, the false positive (negative) rate is 1 minus the true negative (positive) rate. All these metrics range from 0 to 1 and can be interpreted as probabilities – for instance, the true positive rate is the probability that a randomly drawn positive example is correctly classified. In terms of these numbers, the best situation we can have is a true positive rate of 1 (and therefore a false negative rate of 0) and a true negative rate of 1 (and therefore a false positive rate of 0). The true positive rate is sometimes called recall or sensitivity, and the true negative rate is sometimes called specificity. Another proportion that is often used is

precision: the proportion of positive predictions that are correct (*TP*/*PPos* = *TP*/(*TP*+*FP*)).

In its most common form, ROC space is the two-dimensional co-ordinate system with false positive rate on the X-axis and true positive rate on the Y-axis (*Figure 7-1*). Each point in ROC space fixes two of the three degrees of freedom of the contingency table. The remaining degree of freedom is the class distribution (e.g., the number of positives divided by the number of negatives, or by the total number of examples). It makes sense to ignore the class distribution, because it may not be representative, or because different classifiers have been trained with different class distributions. In a way, this is the whole point of ROC analysis, but there are other possibilities. For instance, in information retrieval one wants to ignore the true negatives, which in the case of a search engine would be the number of non-answers that are not returned – we don't really care whether there are 5 thousand or 500 million of those. Ignoring the true negatives can be achieved by using precision (the proportion of positive predictions that are correct) instead of the false positive rate, leading to so-called *precision-recall* diagrams. Globally speaking precision-recall analysis has a similar purpose as ROC analysis, namely to study the operating characteristics of different search engines, but the specifics of the analysis are different because the degree of freedom being ignored is different.

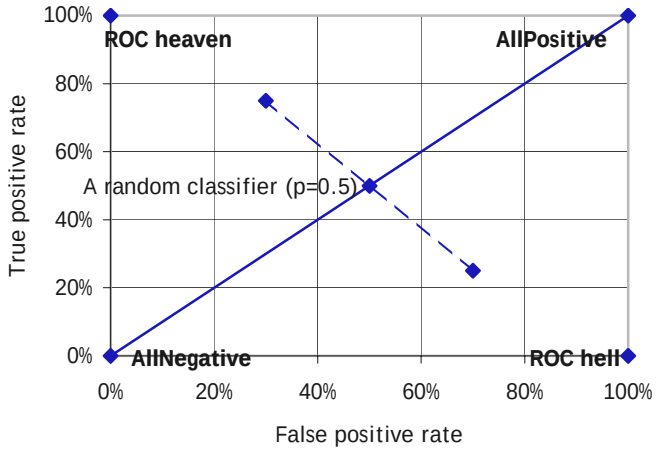

Figure 7-1. Classifiers in ROC space.

We will now have a look at a few special points in ROC space. The origin (0,0) represents a true positive rate of 0 (i.e., all positives are classified as negative) and a false positive rate of 0 (i.e., all negative are classified as negative). In other words, (0,0) represents the classifier which predicts all instances to be negative; we'll call this the *AllNegative* classifier. Analogously, (1,1) represents the *AllPositive* classifier. Jointly, these are called the default classifiers. Note that in ROC analysis, the choice of positive and negative class is irrelevant and has nothing to do with the

majority class. Hence while these two extreme classifiers resemble the decision to classify all examples based on the majority class, these classifiers do not represent such a solution. Moreover, in order to know which of the two default classifiers is the majority class classifier requires knowledge of the class distribution, which we do not have since it has been deliberately factored out.

The point (0,1) corresponds to a true positive rate of 1 (so all positives have been correctly classified) and a false positive rate of 0 (so no negatives are incorrectly classified). In other words, the top left-hand corner represents the classifier that gets it all right. We will sometimes call this point *ROC heaven*, since this is the best possible place to be in ROC space. Analogously, the point (1,0) is the worst possible place: *ROC hell*. However, there is a very easy way to move from hell to heaven: simply flip all predictions from positive to negative, and vice versa. In ROC analysis, it is common to change a given classifier into another one by manipulating its predictions, either deterministically or stochastically – *ROC analysis does not evaluate learning algorithms, but the classifiers they produce*. A learning algorithm that yields the ROC hell classifier is a pretty lousy learning algorithm; but the ROC hell classifier itself is actually not bad at all, since it can be turned into the ROC heaven classifier by a simple trick.

Now take a look at the two default classifiers again, and consider the diagonal connecting them (the positive diagonal). Any point on the diagonal represents a certain true positive rate p and an equal false positive rate. Such behavior can be achieved by a random classifier, which randomly predicts an example to be positive with probability p and negative with probability $(1–p)$. Random classifiers can be constructed without inspecting the dataset at all, i.e., without training. This represents a very useful baseline for learning algorithms, because a learning algorithm is no good if it doesn't result in a classifier above the positive diagonal. Remember, however, that a classifier below the diagonal can easily be transformed in one above the diagonal: for instance, the point (0.7,0.25), i.e., a classifier which correctly classifies half of the negatives but only one quarter of the positives, can be transformed to the point (0.3,0.75) by inverting all predictions. Technically speaking, inverting predictions corresponds to point-mirroring the original point through (0.5,0.5) in ROC space.

Another point worth noting is that, while a random classifier can be seen as making random predictions, it can equally be seen as choosing one of the default classifiers (the extreme points on the positive diagonal) at random. This is a useful perspective because it can be applied to any two points in ROC space. Thus, given two classifiers any behavior on their connecting diagonal can be achieved by making a weighted random choice between the given classifiers for each example to be classified. So, in order to obtain the midpoint between two classifiers we randomly choose between them with equal probability. But even if we do not care about random combinations of classifiers, the connecting diagonal is important for another reason. Consider three classifiers such that the third classifier is below the diagonal connecting the first two. In that case, the third classifier can never outperform both the first and the second, not even if we change the misclassification costs. This can easily be generalized to arbitrary numbers of classifiers, leading to a key concept in ROC analysis: the construction of the ROC convex hull.

3. ROC ANALYSIS FOR MODEL SELECTION

The convex hull of a set of points in ROC space is a piecewise linear curve connecting a selection of points such that all other points are below it – the curve is a *hull*. The resulting curve will not have any 'dents', i.e., each line segment has a slope not steeper than the previous segment – it is *convex*. The convex hull can easily be constructed as follows: starting with (0,0), find the point so that the connecting line segment will be steepest, and continue from that point until you reach (1,1). Computationally this is comparable to sorting n items and hence has a complexity of O(n log n). *Figure 7-2* shows an example of a convex hull.

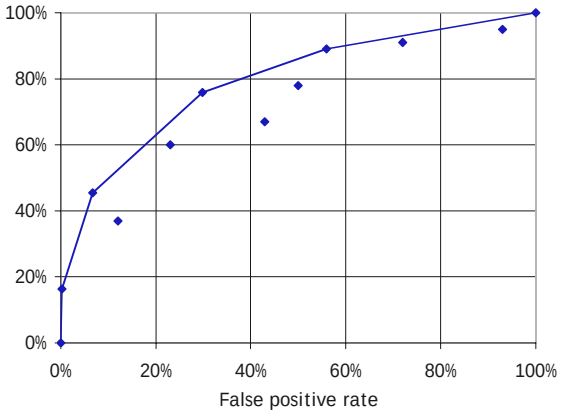

Figure 7-2. The ROC convex hull.

The classifiers on the convex hull are the best classifiers of the entire group, the others can be discarded because they will always be outperformed. The best convex hull possible is the 'curve' formed by the three points (0,0)–(0,1)–(1,1), so the larger the area under the curve, the better it is. If some points we start with are below the diagonal, the procedure just sketched will ensure that they never end up on the convex hull (if all points are below the diagonal, our ROC convex hull will simply be the diagonal itself). In some situations it will be useful to form a separate negative convex hull from the points below the diagonal, e.g., in subgroup discovery. In other situations it may be useful to point-mirror all points below the diagonal through (0.5,0.5), so that some of the mirrored points may actually end up on the (positive) convex hull.

The ROC convex hull is sometimes called a ROC curve, but the latter is a more general concept. A ROC curve similarly connects (0,0) and (1,1) through a number of intermediate points, but it is not necessarily convex (it is monotonically non-decreasing, i.e., no line segment has a negative slope). There are various reasons why a ROC curve may be non-convex, for instance when we re-evaluate the chosen classifiers on a new test set, or when a curve is constructed by setting different thresholds on a probabilistic classifier.

The ROC convex hull is a simple method to select the best ones among a set of given classifiers characterized by their true and false positive rates on a given dataset. If one wants to select a single classifier, one has to know the operating characteristics, i.e., class and cost distributions. From left to right (or from bottom to top, because of convexity) the classifiers on the convex hull become progressively less accurate on the negatives and more accurate on the positives. In order to decide which classifier to use, we need to know the class distribution in the test set (i.e., the context where we want to use the classifier). Suppose the class distribution is 50-50, i.e., equal amounts of positives and negatives. This means that an increase of x in the true positive rate while keeping the false positive rate constant will give the same improvement on the test set as a decrease of x in the false positive rate while keeping the true positive rate constant. So if we draw an imaginary line with slope 1 (i.e., parallel to the diagonal) somewhere in ROC space, all points on that line represent the same test set accuracy. All we need to do is to slide this line in the northwest direction towards ROC heaven, until it touches the ROC curve in a single point: this will be the optimal classifier on the test set. Equivalently, we can select the two line segments with slope closest to 1 (i.e., a slightly steeper segment followed by a slightly less steep segment) and choose the point connecting them.

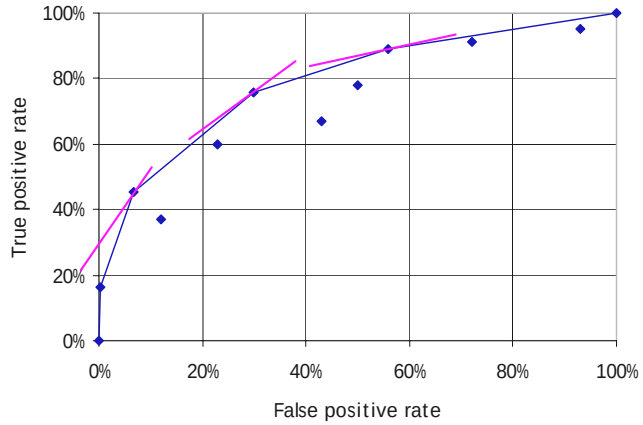

Figure 7-3. Selecting the optimal classifier under given operating characteristics.

The same procedure can be followed for other class distributions: e.g., if we have twice as many negatives in the test set as positives we use a line segment with slope 2, which will force us towards the lower left-hand corner; if we have three times as many positives as negatives, we use a line segment with slope 1/3, and so on. This can be very easily generalized to include misclassification costs: for instance, if we have two times more negatives than positives but false negatives are 10 times as expensive as false positives, we use a slope of 1/5, which naturally selects a classifier that is more accurate on positives than it is on negatives. In general, the slope of a line of equal cost is (C_{FP} / C_{FN}) (*Neg* / *Pos*) with *Pos* and *Neg* as defined above, and C_{FP} / C_{FN} denoting the cost ratio of false positives versus false

negatives. Class distribution and misclassification costs in the test set are combined to obtain the slope of the tangent to the convex hull that will select the optimal classifier. (If it happens that a line segment has exactly the desired slope, we can choose the classifier on either end, or a random combination of them as discussed earlier.) *Figure 7-3* shows a number of different slopes and the classifiers that are selected for these slopes.

The previous analysis is not restricted to models that are classifiers. For instance, in subgroup discovery we are looking for subgroups of the population that have a significantly different target distribution (e.g., subgroups of patients for which a particular therapy's success rate is much higher than average). ROC space is appropriate for measuring the success of subgroup discovery, since subgroups whose *TP*/*FP* tradeoff is close to the diagonal can be discarded as insignificant. Conversely, significant subgroups are those sufficiently distant from the diagonal. In (Lavrač, et al., 2002) the *weighted relative accuracy* metric was proposed for measuring the significance and interestingness of subgroups. The significant subgroups define the points in the ROC space from which a convex hull can be constructed. In (Flach and Gamberger, 2001) this convex hull was used to select the best subgroups in a practical study aimed at target selection for a direct mailing marketing campaign. In order to make decisions easier and to show the expected profit explicitly, ROC curves were transformed into normalized profit curves; we refer the reader to Chapter 8 for details.

4. ROC ANALYSIS FOR MODEL CONSTRUCTION

So far, we have assumed that a fixed set of classifiers is given, for which a ROC convex hull is then constructed. Each of these classifiers is typically obtained by using some off-the-shelf learning algorithm. Most of these algorithms try to optimize some performance criterion such as predictive accuracy. Thus, we are faced with the fact that while we really want to obtain a good ROC convex hull, the classifiers for which we construct this hull are constructed based on entirely different objectives. The question naturally arises whether we can adapt learning algorithms, explicitly incorporating into them the objective of generating (sets of) classifiers that will have good ROC performance.

A first step in this direction was made by (Blockeel and Struyf, 2002), who noted that a decision tree can equally be seen as a set of models by varying the labeling of its leaves. They reason as follows. A tree leaf can be interpreted as predicting positive or negative with a degree of certainty that is related to the class distribution in that leaf. For instance, if the leaf contains 90% positives and 10% negatives, it is reasonable for a new instance belonging to this leaf to be predicted positive, but if the instance had belonged to a leaf with 99% positives we would have been more certain of this prediction. Now, assume that class distribution or misclassification costs are such that we are taking a higher risk when predicting positive than when predicting negative, then we want to predict positive only when we are very sure, e.g., when the instance belongs to a leaf with at least 95% positives. Thus, from a given decision tree we can derive a set of decision trees, starting with a tree that predicts negative everywhere and gradually introducing

positive predictions in those leaves that have the next highest proportion of positives, until for the final tree all leaves predict positive. We say that the original tree has no bias towards positives or negatives (it is optimal for symmetric misclassification costs and class distributions), whereas the derived trees are biased to a different extent towards positive or negative predictions.

In an experimental evaluation of this procedure, a comparison was made between the convex hull of (a) a set of classifiers built using various procedures, one of which was a decision tree, and (b) the same set of classifiers extended with classifiers derived from the tree. A clear improvement of the ROC convex hull was obtained in the latter case (Blockeel and Struyf, 2002).

(Ferri, et al., 2002) take this idea further in two different directions. First, they provide a theoretical motivation for the procedure just described. They consider all possible re-labelings of a tree instead of just those considered by Blockeel and Struyf. This gives a set of 2^n different trees, for which they next prove that the convex hull consists of $n+1$ specific labelings, exactly those used by Blockeel and Struyf.

Second, they extend the procedure in the following way. Instead of generating a set of trees from a single given tree, which is typically grown using a standard tree induction algorithm, they propose to adapt the tree induction algorithm itself in such a way that a tree is built for which the derived set will have optimal ROC properties, that is, will have a maximal area under the convex hull (AUC). Note how the ROC analysis criterion is in a sense consecutively pushed deeper into the generation of the set of classifiers: whereas ROC analysis assumes a set of classifiers given, and Blockeel and Struyf assume a single classifier given and generate a set from it, Ferri et al. generate this single classifier with the aim of obtaining a good ROC evaluation.

	+	−
LEAF 1	3	5
LEAF 2	5	1
LEAF 3	4	2

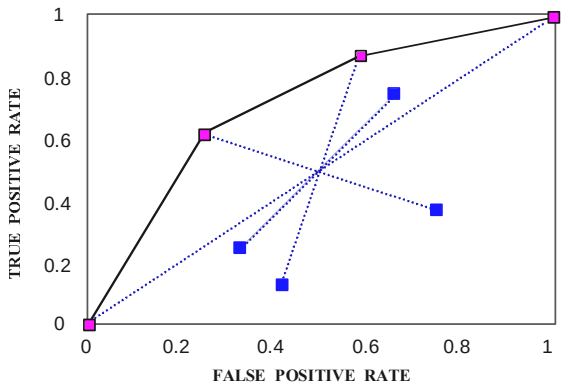

Figure 7-4. A decision tree with all its possible labelings in ROC space.

The following example, taken from Ferri et al., illustrates these issues. Suppose we have a decision tree with *three* leaves and the training set distribution as in *Figure 7-4*, which also depicts the ROC points of the $2^n=8$ (where n is the number of leaves in the tree) possible labelings As can be seen in the figure, the points are

mirrored through the point (0.5, 0.5), because for each labeling there is another labeling assigning the opposite class to each leaf. To obtain the optimal labelings on the convex hull, we order the leaves by their accuracy on the positives, and then we generate the set of n+1=4 optimal labelings (referred to as S_0–S_3) as follows:

	+	–	S_0	S_1	S_2	S_3
LEAF 2	5	1	-	+	+	+
LEAF 3	4	2	-	-	+	+
LEAF 1	3	5	-	-	-	+

The aim of decision tree construction then is to construct a tree that performs well for any of its ROC-optimal labelings. For evaluating the quality of such an unlabelled tree the *area under the ROC curve* (*AUC*) metric is very natural. (Ferri, et al., 2002) use a local version of the AUC metric as a novel splitting criterion for deciding how to grow the tree, and show that it outperforms other splitting criteria both with respect to classification accuracy and area under the ROC curve. The AUC-based splitting criterion is interesting because, unlike traditional splitting criteria (e.g., information gain), it does not perform a comparison between the impurity of the parent node with the weighted impurity of the children after splitting. For instance, in the case of a binary split of a parent with p positives and n negatives into two children with p_1 and p_2 positives and n_1 and n_2 negatives, respectively, the AUC-based splitting criterion evaluates the quality of this split as $(p_1n+pn_2)/2pn$.

5. CONCLUDING REMARKS

In this chapter we have given a general introduction to ROC analysis and its applications in decision support for data mining (model selection) as well as some novel applications in model construction and combination. In our opinion, ROC analysis is an important subject with wide-ranging applications across the board in data mining. Further background on ROC analysis in machine learning and data mining can be found in (Provost and Fawcett, 2001) and (Fawcett, 2003).

We have restricted attention to two-class problems. In a multi-class setting with c classes, a full ROC plot would have $c(c–1)$ dimensions – one dimension for each possible misclassification (class 1 predicted as class 2, as class 3, etc.). This is often approximated by combining all possible misclassifications for a particular class, leading to c dimensions. Recent work includes an algorithm for calculating the convex hull in full ROC space (Srinivasan, 1999) and approximating the area under the ROC curve (Hand and Till, 2001).

REFERENCES

Blockeel, H. and Struyf, J. (2002). Deriving biased classifiers for improved ROC performance, *Informatica*, Vol. 26, No. 1, 77–84.

Fawcett, T. (2003). ROC Graphs: Notes and Practical Considerations for Data Mining Researchers, Hewlett-Packard Laboratories.

Ferri, C., Flach, P. and Hernández-Orallo, J. (2002). Decision tree learning using the area under the ROC curve. *Proc. 19th International Conference on Machine Learning (ICML'02)*. (eds. Sammut, C. and Hoffman, A.), Morgan Kaufmann, 139–146.

Flach, P. and Gamberger, D. (2001). Subgroup evaluation and decision support for a direct mailing marketing problem. *Proc. ECML/PKDD-2001 Workshop Integrating Aspects of Data Mining, Decision Support and Meta-Learning (IDDM-2001)*. (eds. Giraud-Carrier, C., Lavrač, N., Moyle, S. A. and Kavšek, B.), Freiburg, Germany, 45–56.

Hand, D. J. and Till, R. J. (2001). A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems, *Machine Learning*, Vol. 45, No. 2, 171–186.

Lavrač, N., Flach, P., Kavšek, B. and Todorovski, L. (2002). Adapting classification rule induction to subgroup discovery. *Proc. 2002 IEEE International Conference on Data Mining*. IEEE Press, 266–273.

Provost, F. and Fawcett, T. (2001). Robust classification for imprecise environments, *Machine Learning*, Vol. 42, No. 3, 203–231.

Srinivasan, A. (1999). Note on the location of optimal classifiers in n-dimensional ROC space, Oxford University Computing Laboratory.