Mach Learn (2007) 68: 97–106
DOI 10.1007/s10994-007-5011-0

TECHNICAL NOTE

PAV and the ROC convex hull

Tom Fawcett · Alexandru Niculescu-Mizil

Received: 30 November 2006 / Revised: 4 April 2007 /
Accepted: 5 April 2007 / Published online: 15 May 2007
Springer Science+Business Media, LLC 2007

Abstract Classifier calibration is the process of converting classifier scores into reliable probability estimates. Recently, a calibration technique based on isotonic regression has gained attention within machine learning as a flexible and effective way to calibrate classifiers. We show that, surprisingly, isotonic regression based calibration using the Pool Adjacent Violators algorithm is equivalent to the ROC convex hull method.

Keywords Classification · Classifier calibration · ROC · Class skew

1 Introduction

Most binary classification models are commonly used to provide not just class labels but also instance scores. These scores are usually interpreted as the confidence the classifier has in its prediction: the higher the score, the higher the probability that the given case is positive.

In some applications the scores generated by classifiers are used to estimate posterior class membership probabilities. The posterior probabilities are necessary for example when the classifier is used for cost-sensitive applications, in which precise judgments about the cost of errors must be made. However, the raw scores are not always good estimates of true probabilities. Some model classes are notoriously poor at producing accurate estimates (Niculescu-Mizil and Caruana 2005), so before using scores as posterior probability estimates, they must be *calibrated*. Recently, a technique based on isotonic regression has gained attention as a simple and effective way to calibrate classifiers (Zadrozny and Elkan

Editor: Johannes Fürnkranz.

T. Fawcett (✉)
Center for the Study of Language and Information, Stanford University, Stanford, CA 94305, USA
e-mail: tfawcett@acm.org

A. Niculescu-Mizil
Computer Science Department, Cornell University, Ithaca, NY 14853, USA
e-mail: alexn@cs.cornell.edu

2002; Niculescu-Mizil and Caruana 2005; Caruana and Niculescu-Mizil 2006). The calibration technique uses the Pool Adjacent Violators (PAV) algorithm to find an isotonic (monotonically increasing) transformation of the classifier's scores that yields the lowest Brier Score (and cross-entropy).

In other application domains, the scores generated by a classifier do not have to be treated as calibrated posterior probabilities. The scores are instead used for an ROC analysis of the classifier(s). The ROC analysis is useful, for example, to estimate the expected performance of a classifier under varying misclassification costs. In this context, Provost and Fawcett (2001) introduced the ROC convex hull (ROCCH) algorithm. The initial purpose of the ROCCH was to facilitate the selection of the classifier that is optimal for given misclassification costs and the elimination of classifiers that are never optimal regardless of the misclassification costs

In this paper we prove that the PAV and ROCCH algorithms are equivalent, a surprising result given that the two algorithms were developed in different contexts and for different purposes. Indeed, generating the ROCCH of a single classifier can be viewed as finding an isotonic transformation of the scores that maximizes the area under the ROC curve (AUC). This transformation turns out to be isomorphic to the isotonic transformation generated by the PAV algorithm that minimizes Brier Score and cross-entropy.

The purpose of this article is not to argue that one technique subsumes the other, or that one is superior to the other, but to demonstrate the connection between these apparently unrelated techniques. Showing that the two are equivalent allows extensions and insights for one technique to be applied to the other. Furthermore, revealing such connections might lead to new thinking in the community.

2 Calibration and the PAV algorithm

In many applications it is important to be able to interpret the scores output by a classifier as well calibrated posterior probabilities. For example, in cost-sensitive decision making, where different instances have different misclassification costs that might not be known at training time, well calibrated probabilities are essential to making decisions that will minimize the total expected cost. Probabilistic predictions are also essential in domains such as medical decision making, weather forecasting, fraud detection and risk analysis. Even when probabilistic outputs are not really necessary, they are often desirable because they are more meaningful and easier to interpret than some uncalibrated score.

Unfortunately, some widely used classifiers do not make well calibrated probabilistic predictions. For example, SVMs and AdaBoost were not designed to generate probabilistic classifiers, and Naive Bayes is well known to produce biased probability estimates due to unrealistic independence assumptions (Niculescu-Mizil and Caruana 2005). In order to use such classifiers in the above mentioned applications, their predictions must be *calibrated*; i.e. the instance scores must be transformed into well-calibrated posterior probabilities.

Formally, given a classifier f that scores instances, $f : X \to R$, calibration creates a function m such that $m(f(x)) \to [0, 1]$ is an estimate of the posterior probability that instance x is in the positive class. Various methods have been proposed for finding the mapping m such as methods based on parametric estimation (Platt 1999), binning (Zadrozny and Elkan 2001) or nonparametric Isotonic Regression (Zadrozny and Elkan 2002).

Initially introduced by Zadrozny and Elkan (2002), the Isotonic Regression method relies on the assumption that the function that transforms the scores output by a classifier into calibrated posterior probabilities is monotonically increasing (isotonic) with respect to the classifier score; i.e. a higher score implies a higher probability that the instance is positive. Given a training set (f_i, y_i), where $f_i = f(x_i)$ is the output of the classifier for instance x_i and $y_i \in \{0, 1\}$ is the corresponding label, Isotonic Regression will find the isotonic function m such that:

$$m = \arg\min_z \sum (y_i - z(f_i))^2 \tag{1}$$

where the arg min is taken over all isotonic functions. While the Isotonic Regression problem is presented here in terms of Brier Score,[1] it can be proved that the same isotonic transformation is obtained when using any proper scoring function,[2] including cross-entropy (log-loss).

The most extensively studied algorithm for the isotonic regression problem is the Pool Adjacent Violators (PAV) algorithm. The PAV algorithm is conceptually straightforward. Given a set of training cases ordered by the scores assigned by the classifier, it first assigns a probability of one to each positive instance and a probability of zero to each negative instance, and puts each instance in its own group. It then looks, at each iteration, for adjacent *violators*: adjacent groups whose probabilities locally increase rather than decrease. When it finds such groups, it pools them and replaces their probability estimates with the average of the group's values. It continues this process of averaging and replacement until the entire sequence is monotonically decreasing. The result is a sequence of instances, each of which has a score and an associated probability estimate, which can then be used to map scores into probability estimates.

The pseudocode for the PAV algorithm is shown in Algorithm 1. This is a conceptual rendition of the algorithm meant to better illustrate the equivalence with the ROCCH algorithm; an efficient version of the algorithm which has $O(N)$ time and space complexity is given by Wilbur et al. (2005).

Algorithm 1 Basic PAV method for generating probability estimates

Input: Scored training set (f_i, y_i), where f_i is the score assigned by the classifier and y_i is the correct class.

Output: Stepwise constant function generated by m

```
begin
Sort training set instances increasing by $f_i$
Put each training instance in its own group, $G_{i,i}$ and predict $m_{i,i} = y_i$
while $\exists G_{k,i-1}$ and $G_{i,l}$ ST $m_{k,i-1} \geq m_{i,l}$ do
    Pool the instances in $G_{k,i-1}$ and $G_{i,l}$ into one group, $G_{k,l}$
    $m_{k,l} = (\sum_{i=k}^{l} y_i)/(l - k + 1)$
    Predict $m_{k,l}$ for all instances in $G_{k,l}$
end while
Output the stepwise constant function generated by $m$
end
```

Table 1 shows an example of the PAV algorithm operating on a sequence of 15 instances, six negatives and nine positives. The PAV algorithm begins by sorting the instances in decreasing order by score and assigning probability estimates of 1 for each positive example

[1] Brier Score is used to assess the quality of the probabilities predicted by a classifier (Brier 1950). It is equivalent to squared error for two-class problems.

[2] A scoring rule is said to be proper if it is minimized (maximized) in expectation when the true posterior probability is predicted.

Table 1 An illustration of the PAV algorithm

#	Score	Probabilities						
		Initial	a1	a2	b	c1	c2	d
0	0.9	1	1	1	1	1	1	1
1	0.8	1	1	1	1	1	1	1
2	0.7	0	0	0	0	0	0	3/4
3	0.6	1	1	1	1	1	1	3/4
4	0.55	1	1	1	1	1	1	3/4
5	0.5	1	1	1	1	1	1	3/4
6	0.45	0	0	0	0	1/2	2/3	2/3
7	0.4	1	1	1	1	1/2	2/3	2/3
8	0.35	1	1	1	1	1	2/3	2/3
9	0.3	0	0	0	1/2	1/2	1/2	1/2
10	0.27	1	1	1	1/2	1/2	1/2	1/2
11	0.2	0	0	1/3	1/3	1/3	1/3	1/3
12	0.18	0	1/2	1/3	1/3	1/3	1/3	1/3
13	0.1	1	1/2	1/3	1/3	1/3	1/3	1/3
14	0.02	0	0	0	0	0	0	0

and 0 for each negative example. The algorithm iteratively looks for adjacent violators: a local non-monotonicity in the sequence. Initially, adjacent violators (a zero followed by a one) exist at instance pairs 2–3, 6–7, 9–10 and 12–13.

We will describe the algorithm operating from the bottom of the instance sequence to the top. First, in step a1, the violation generated by instances 12 and 13 is removed by pooling the two instances together and assigning them a probability estimate of $1/2$ (see column a1). This introduces a new violation between the instance 11 and the adjacent group 12–13. To remove this new violation, in step a2, instance 11 and the group 12–13 are pooled together, forming a pool of three instances (two negatives and one positive) whose probability estimate is $1/3$. The result is shown in column a2.

Next, instances 9–10 (one negative and one positive) are pooled, assigning a probability of $1/2$ to each instance. The result is shown in column b.

In steps c1 and c2, the violations between instances 6–8 (one negative and two positives) are removed in two steps. Similarly, instances 2–5 (one negative and three positives) are pooled into a group of probability $3/4$ (the intermediate steps are omitted). The final result is shown in column d. The sequence of probability estimates is now monotonically decreasing and no violators remain. This sequence can now be used as the basis for a function that maps classifier scores into probability estimates.

3 ROC space and the ROC convex hull algorithm

Receiver Operating Characteristic (ROC) analysis was first employed during the World War II to analyze the predictions made by RADAR "receiver operators" (Swets 1988; Swets et al. 2000). ROC analysis has subsequently become a classic tool in signal detection theory and it has been widely used in medical diagnosis, psychology, radiology, and, more recently, machine learning.

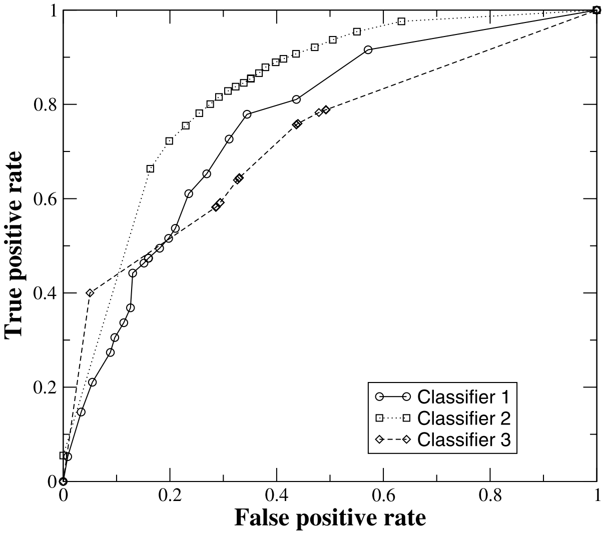

Fig. 1 An ROC graph of three scoring classifiers

Let $\{\mathbf{p}, \mathbf{n}\}$ be the positive and negative instance classes, and let $\{\mathbf{Y}, \mathbf{N}\}$ be the classifications produced by a hard classifier. The true positive rate, *tp rate*, of a classifier is:

$$\textit{tp rate} = p(\mathbf{Y}|\mathbf{p}) \approx \frac{\text{positives correctly classified}}{\text{total positives}}.$$

The false positive rate, *fp rate*, of a classifier is:

$$\textit{fp rate} = p(\mathbf{Y}|\mathbf{n}) \approx \frac{\text{negatives incorrectly classified}}{\text{total negatives}}.$$

The *tp rate* and *fp rate* of a hard classifier define the coordinates of a point in *ROC space*, with *tp rate* plotted on the Y axis and *fp rate* plotted on the X axis. When a classifier outputs continuous scores, a hard classification may be obtained by using a threshold to decide the cutoff point. The *tp rate* and *fp rate* statistics vary together monotonically as the threshold is varied from the lowest output score to the highest, resulting in a curve called the ROC curve. An ROC curve illustrates the error trade-offs made by a given classifier under all misclassification costs and all class distributions. Figure 1 shows a typical ROC plot of three classifiers.

Algorithm 2 is a basic algorithm for generating an ROC curve from a data set. It exploits the monotonicity of thresholded classifications: any instance that is classified as positive with respect to a given threshold will be classified as positive for all lower thresholds. This algorithm assumes that the classifier assigns scores to instances. The function $f(i)$ is the score assigned to instance i by the classifier. In this algorithm, *tp rate* and *fp rate* start at zero. Each positive instance increments *tp rate* by $1/\text{total positives}$ and each negative instance increments *fp rate* by $1/\text{total negatives}$. The algorithm maintains a stack R of ROC points, pushing a new point onto R after each group of tied instances (instances with the same score) is processed. The final output is the stack R, containing the points on the ROC curve. Each group of tied instances will therefore generate on the ROC graph a line segment with slope equal to:

$$\text{slope} = \frac{\text{pos}/\text{total positives}}{\text{neg}/\text{total negatives}} = \frac{1}{\text{skew}} \times \frac{\text{pos}}{\text{neg}} \tag{2}$$

where pos and neg represent the number of positive cases and negative cases respectively in the group, and skew is the ratio of positive to negative cases in the data set.

Algorithm 2 Generating an ROC curve

```
Inputs: $L$, the set of test examples; $f(i)$, the probabilistic classifier's estimate that
example $i$ is positive; $P$ and $N$.
Outputs: $R$, a list of points defining the ROC curve.
begin
$L_{\text{sorted}} \leftarrow L$ sorted decreasing by $f$ scores
$\textit{fp rate} \leftarrow 0$; $\textit{tp rate} \leftarrow 0$
$R \leftarrow \langle\rangle$
$f_{\text{prev}} \leftarrow -\infty$
$i \leftarrow 1$
while $i \leq |L_{\text{sorted}}|$ do
  if $f(i) \neq f_{\text{prev}}$ then
    push $(\textit{fp rate}, \textit{tp rate})$ onto $R$
    $f_{\text{prev}} \leftarrow f(i)$
  end if
  if $L_{\text{sorted}}[i]$ is a positive example then
    $\textit{tp rate} \leftarrow \textit{tp rate} + 1/P$
  else                                        /*i is a negative example*/
    $\textit{fp rate} \leftarrow \textit{fp rate} + 1/N$
  end if
  $i \leftarrow i + 1$
end while
push $(1, 1)$ onto $R$
end
```

Provost and Fawcett (2001) show that a classifier is potentially optimal if and only if it lies on the convex hull of the set of points in ROC space. The convex hull of the set of points in ROC space is called the *ROC convex hull* (ROCCH) of the corresponding set of classifiers. The ROCCH is a piecewise-linear, concave-down "curve." Therefore, as x increases, the slope of the ROCCH is monotonically non-increasing with $k - 1$ discrete values, where k is the number of ROCCH component classifiers, including the degenerate classifiers that define the hull endpoints.

In its original formulation, the ROCCH method does not generate probabilities from instance scores. However, each point on the ROC graph corresponds to a score, and (2) can be used to map hull slopes into probability estimates once the skew is known. Section 4 explains this.

The ROCCH formulation has a number of useful implications. Since only the classifiers on the convex hull are potentially optimal, no others need be retained. The conditions under which a classifier should operate (class skew and error costs) may be translated into a so-called iso-performance line, which in turn may be used to identify a portion of the ROCCH containing the optimal classifiers for those conditions.

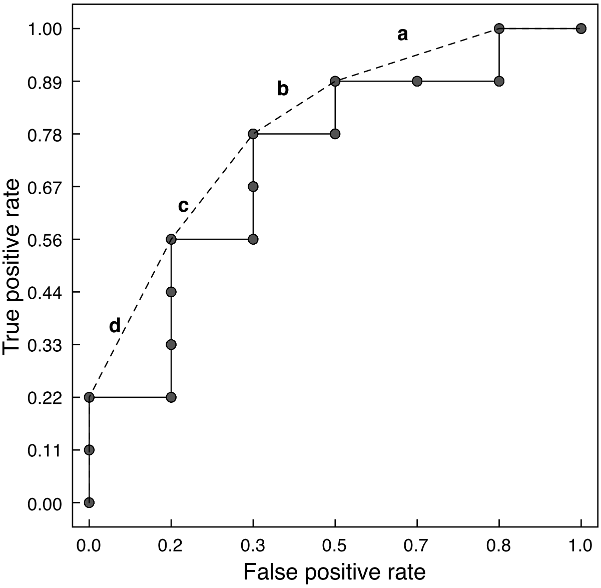

Fig. 2 The instances of Table 1 in ROC space

Algorithm 3 ROC Convex Hull Algorithm for a single classifier

Input: Scored training set (f_i, y_i), where f_i is the score assigned by the classifier and y_i is the correct class.

Output: Stepwise constant function generated by r

```
begin
Sort training set instances increasing by $f_i$
Put each training instance in its own group, $G_{i,i}$
Generate the ROC curve
while $\exists G_{k,i-1}$ and $G_{i,l}$ that generate a concavity in the ROC curve do
    Pool the instances in $G_{k,i-1}$ and $G_{i,l}$ into one group, $G_{k,l}$
    Let $r_{k,l}$ ST $f_k \leq r_{k,l} \leq f_l$
    Predict $r_{k,l}$ for all instances in $G_{k,l}$
    Regenerate the ROC curve
end while
Output the stepwise constant function generated by $r$
end
```

Figure 2 shows the ROC graph and its convex hull for the example in Table 1. The solid line represents the ROC graph and the dashed line the convex hull. Each hull segment can be viewed as being generated by a tie between the instances it "covers". In fact, in the case of a single classifier, the ROC convex hull can be viewed as generating an isotonic transformation of the original classifier intended to maximize the area under the ROC curve. To achieve this, the ROCCH algorithms "fixes" the concavities in the ROC graph by grouping instances that generate a concavity and creating a tie between them (i.e. predicting the same value for all instances in the group). The pseudocode for generating the ROC convex hull transformation for a classifier is shown in Algorithm 3. Again, this is a rather inefficient rendition of the algorithm that is only intended to illustrate the connection with the PAV algorithm.

Note that each convex hull segment in Fig. 2 corresponds to a pooling of the instances performed in Sect. 2, Table 1. For example, the line segment labeled d corresponds to pooling of instances 2–5, resulting in a group comprising one negative and three positive examples; the segment labeled c corresponds to grouping instances 6–8 and so on. Each hull segment covers a group of instances with the same prediction generated by the PAV algorithm, and each group of instances generated by PAV is covered by a hull segment. So, for this example, the PAV and ROCCH algorithms find the same isotonic transformation. In the next section we prove that this equivalence holds for all classifiers.

4 Equivalence between PAV and ROCCH

In this section we prove the equivalence between the PAV algorithm and the ROCCH algorithm for a single classifier. Formally we prove that:

Theorem 1 *For any binary classifier and for any training set* (f_i, y_i) *the following are true*:

1. *The ROC graph of the PAV-transformed classifier is identical to the convex hull of the original classifier.*
2. *For each instance, the prediction made by the PAV-transformed classifier is equal to*

$$\frac{\text{slope} \cdot \text{skew}}{1 + \text{slope} \cdot \text{skew}}$$

where slope is the slope of the segment on the convex hull of the original classifier the instance belongs to and skew is the ratio of positive to negative cases in the training set.

Proof To prove the theorem we will use the versions of the PAV and single-classifier ROCCH presented in algorithms 1 and 3 respectively. Both algorithms work by grouping adjacent instances in order to remove violations. In the case of PAV the violation consists of a local non-monotonicity, while in the case of ROCCH a violation consists of a local concavity in the ROC curve.

The order in which adjacent groups are joined has no affect on the final result for either of the two algorithms. Indeed, for any order, the PAV algorithm will find the maximum likelihood (minimum cross-entropy) solution which is unique (Ayer et al. 1955). Similarly, for any order, the ROCCH algorithm will find the convex hull of the ROC curve which is also unique.

Since both algorithms start with the same initial groups (i.e. each point in its group), and the final result is independent of the order in which groups are merged by the two algorithms, it is sufficient to prove that whenever one algorithm decides to join two groups the other algorithm will join them too. So we only have to prove that two adjacent groups will generate a concavity in the ROC curve if and only if they will generate a violation.

Let $G_{k,j-1}$ and $G_{j,l}$, with $f_k \leq f_{j-1} \leq f_j \leq f_l$ be two groups that generate a concavity in the ROC curve. Let $\text{slope}_{k,j-1}$ and $\text{slope}_{j,l}$ be slopes of the segments corresponding to $G_{k,j-1}$ and $G_{j,l}$ respectively. Let

$$pos_{s,t} = \sum_{i=s}^{t} y_i,$$

$$neg_{s,t} = \sum_{i=s}^{t} 1 - y_i$$

be the number of true positive cases and true negative cases in the group $G_{s,t}$. If $G_{k,j-1}$ and $G_{j,l}$ generate a concavity on the ROC curve (slope$_{k,j-1}$ > slope$_{j,l}$), then:

$$\frac{pos_{k,j-1}}{neg_{k,j-1}} = \text{slope}_{k,j-1} \cdot \text{skew} > \text{slope}_{j,l} \cdot \text{skew} = \frac{pos_{j,l}}{neg_{j,l}}.$$

This is true if and only if:

$$m_{k,j-1} = \frac{\sum_{i=k}^{j-1} y_i}{j-k} = \frac{pos_{k,j-1}}{pos_{k,j-1} + neg_{k,j-1}} > \frac{pos_{j,l}}{pos_{j,l} + neg_{j,l}} = m_{j,l}$$

which means that $G_{k,j-1}$ and $G_{j,l}$ generate a violation in the PAV algorithm.

In conclusion, both algorithms generate the same groups of tied instances implying that the ROC graph of the PAV-transformed classifier is the convex hull of the original classifier.

The relationship between the prediction made by the PAV-transformed classifier for an instance and the slope of convex hull segment that instance belongs to follows directly from (2). □

5 Conclusions

In this paper we prove that the pool adjacent violators algorithm and the ROC convex hull algorithm are equivalent. Essentially, PAV and ROCCH converge to the same transformation from different directions. ROCCH always maintains the order of the cases and improves the performance at each step, while PAV starts with the best possible classifier and, at each step, corrects the ordering.

An interesting corollary is that for any classifier there exists a single isotonic transformation that yields, for a given data set, optimal ROC area, Brier Score, cross-entropy, and minimum cost for any choice of threshold value and misclassification costs.

Showing the equivalence between the two techniques allows extensions and insights for one technique to be applied to the other. For example, Flach and Wu (2005) have shown how classifier performance may be improved by rearranging instance scores so as to eliminate concavities in ROC space. Because of the isomorphism between ROC space and the PAV algorithm, their technique may be applied to the PAV algorithm as well to improve probability scores. We hope that this article stimulates new thinking among machine learning researchers of both probability estimation and ROC techniques, and leads to improvements to each group of methods.

Acknowledgements Discussions with Rich Caruana prompted us to investigate the relationship between PAV and the ROCCH. Niculescu-Mizil was supported by NSF Award 0412930.

References

Ayer, M., Brunk, H., Ewing, G., Reid, W., & Silverman, E. (1955). An empirical distribution function for sampling with incomplete information. *Annals of Mathematical Statistics*, *5*(26), 641–647.

Brier, G. W. (1950). Verification of forecasts expressed in terms of probabilities. *Monthly Weather Review*, *78*, 1–3.

Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In W. Cohen & A. Moore (Eds.), *Proceedings of the twenty-third international conference on machine learning (ICML'06)* (pp. 161–168). New York: ACM.

Flach, P. A., & Wu, S. (2005). Repairing concavities in ROC curves. In L.P. Kaelbling & A. Saffiotti (Eds.), *Proceedings of the nineteenth international joint conference on artificial intelligence (IJCAI'05)* (pp. 702–707). Berlin: Springer.

Niculescu-Mizil, A., & Caruana, R. (2005). Predicting good probabilities with supervised learning. In L.D. Raedt & S. Wrobel (Eds.), *Proceedings of the twenty-second international conference on machine learning (ICML'05)* (pp. 625–632). New York: ACM.

Platt, J. (1999). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schoelkopf & D. Schuurmans (Eds.), *Advances in large margin classifiers* (pp. 61–74). Cambridge: MIT.

Provost, F., & Fawcett, T. (2001). Robust classification for imprecise environments. *Machine Learning*, *42*(3), 203–231.

Swets, J. (1988). Measuring the accuracy of diagnostic systems. *Science*, *240*, 1285–1293.

Swets, J. A., Dawes, R. M., & Monahan, J. (2000). Better decisions through science. *Scientific American*, *283*, 82–87.

Wilbur, W. J., Yeganova, L., & Kim, W. (2005). The synergy between PAV and AdaBoost. *Machine Learning*, *61*(1–3), 71–103.

Zadrozny, B., & Elkan, C. (2001). Learning and making decisions when costs and probabilities are both unknown. In F. Provost & R. Srikant (Eds.), *Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining (KDD'01)* (pp. 204–213).

Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. In D. Hand, D. Keim & R. Ng (Eds.), *Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining (KDD'02)* (pp. 694–699). New York: ACM.