Archived at the Flinders Academic Commons: http://dspace.flinders.edu.au/dspace/

This is the initial version of a paper submitted to the 11th International Conference on Informatics in Control, Automation and Robotics, 1-3 September 2014, Vienna, Austria.


Copyright © 2013 The author.

Paper reproduced here with permission from the author.
Abstract: There has been considerable interest in boosting and bagging, including the combination of the adaptive Boosting is a technique for turning a weak learner into a strong learner in terms of Valiant’s (1984, 1989) Probably Approximately Correct framework, where a strong learner is defined as being arbitrarily close to perfect and a weak learner is defined as being marginally better than chance, where the performance of the algorithms is limited as a polynomial of the reciprocals of the arbitrarily small deviations from perfection or chance respectively. However, Shapire (1999) original algorithm and proof for boosting only considered the dichotomous (two-class) case and made the assumption that chance level performance was 1/2, on the basis that guesses are unbiased coin tosses. Practical boosting algorithms (Freund, 1995) followed based on the same idea an iteratively applied weak learner, concentrating on the examples which were not classified correctly. Adaptive boosting (Freund & Shapire, 1997), Adaboost, used weights on instances for the next training of the weak learner were adjusted according to the odds \( \frac{1}{a} \) of being correct in order to even up the score and force finding a new way of making an above chance decision, where a is the accuracy (proportion right) and \( a < 1 \) is the error (proportion wrong). In returning the composite classifier, a linear weighting using the log odds is used: \( \text{ln}(\frac{a}{1-a}) \). While \( \frac{1}{a} < 1 \) boosting can continue – otherwise the final classifier is built: equality with 1 means that the weak learner returned a perfect result and the stronger learner has been successfully achieved, while equality with \( \frac{1}{2} \) means that a chance level score was achieved and the weak learner has failed.

Many generalizations exist to the multiclass case (Shapire & Singer, 2000), including a variant on the dichotomous algorithm that simply used a K-class accuracy function. There are two practical considerations that make these two boosting algorithms (and the whole family of boosting algorithms based on error correcting output codes) unsuited for signal processing: they both require a collection of error probability-like measures, and we illustrate these in terms of counts of various conditions or contingencies: 

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}
\]

1. APPLICATION IN SENSOR, TEXT & SIGNAL PROCESSING

Whereas the preceding discussion has been in a general Machine Learning context we take a moment to bring the discussion to the practical level and discuss the kinds of applications and learning algorithms where the proposed techniques can make a large difference. The practical context of the work reported here, including the development and testing of the various chance-corrected measures, is signal processing of Electroencephalographic Brain Computer Interface experiments (Finggibbon et al., 2007, 2013; Ataybi et al., 2013), Audiovisual Speech, Gesture, Expression and Emotion Recognition (Lewis & Powers, 2004; Jia et al., 2012, 2013) and Information Retrieval and Language Modelling (Powers, 1983, 1991; Yang & Powers, 2006; Huang & Powers, 2001) and it was in this Natural Language Processing context that the problems of evaluation and its role in the misleading of learning systems were first recognized (Entwisle & Powers, 1998). Furthermore, although the work reported here uses standard datasets, we use character or text data sets that pertain to this natural language task because the problem we are identifying, and the advantage of solving it, grows with the number of classes. Similarly the software used is modifications of standard algorithms as implemented in Weka (Witten et al., 2011), so that comparison with other multiclass boosting work is possible. Two of Weka’s Boosting algorithms, AdaBoostM1 and Multibook (Wolth, 1996), are used as the basis for the proposed modifications, with Tree-based and Perceptron-based learners preferred to stable learners, like Naïve Bayes, that don’t boost (Fig. 1).

There are two practical considerations that make these two boosting algorithms (and the whole family of boosting algorithms based on error correcting output codes) unsuited for signal processing: they both require a collection of error probability-like measures, and we illustrate these in terms of counts of various conditions or contingencies: 

\[
x = [\text{all}\text{-}1][\text{all}\text{-}0] = \text{TruePositives}/\text{RealPoses}
\]

(1)

\[
\text{Accuracy} = \text{TruePositives}/\text{PredicPoses}
\]
The expected accuracy is defined differently depending on the particular method of chance correction that is used. For example, the common method is to subtract the expected performance, and the improved accuracy is computed as follows: \( E_{acc}(x) = \frac{1}{2} \) for any chance-corrected Kappa \( x \), and we define the associated Error as 1 – Accuracy

\[ E_{acc}(x) = \text{PredictedPos/AllCases} - \frac{1}{2} \]

\[ E_{acc}(x) = \text{RecallPos/AllCases} - \frac{1}{2} \]

An Adaboost and many other boosting algorithms are defined in terms of Rand Accuracy or equivalently the proportion of Error, and can thus be straightforwardly adapted by substituting the corresponding alternate definition. Our prediction is that optimising Cohen’s Kappa and Powers’ Bookmaker Informedness are expected to do far better than the uncorrected Rand Accuracy (or proportional Error) or other tested measures including Powers’ Markedness and Matthews’ Correlation, when tested in Weka's implementation (Witten et al., 2011) of Adaboost.M1 using tree stumps/learners as the weak learners.

Informedness is expected to perform best when the weak learner is unbiased or prevalence-biased, but sometimes Kappa can be expected to be better, in particular, when the weak learner optimizes Kappa or Accuracy, which is linearly related to Kappa for a fixed estimate of the expected accuracy (which for Cohen Kappa corresponds to fixed marginal probabilities). Kappa can go up and Informedness down, when the predictive bias (proportion of predictions) for a particular label varies from population prevalence (proportion of real labels) (Powers, 2012). No learners that explicitly optimize Informedness are known, but all learners that match Label Bias to Class Prevalence will maximize all forms of Kappa and Correlation, including Informedness, whatever form of error they minimize or accuracy they maximize subject to that constraint. This has long been a heuristic for the setting of thresholds in neural networks, and can also be used in Receiver Operating Characteristics (ROC) optimization. However in ROC analysis this corresponds with intersecting the fn=fp diagonal and is not in general the optimum operating point. Mismatching Bias and Prevalence can lead to gains over equal Bias and Powers (Powers, 2011).

Moreover, the base learners originally used with Adaboost were tree-type learners, and in a leaf node these algorithms can be expected to make the majority decision or an equiprobable guess. The former seems to be unbiased and the latter by the locally conditioned prevalences of that node rather than the global population prevalences that are appropriate for optimizing a chance-corrected measure, and this in particular is inappropriate for Informedness. On the other hand, neural-based learners, and Bayesian learners, do not have such a simple majority voting bias. Moreover, Adaboost as a convex learner has strong similarities to neural networks and SVM, but a Naïve Bayes learner could provide a quite distinct behaviour and provide a weak learner that also satisfies the requirement of being a fast learner.

Note that if a weak learner doesn’t have the local majority bias of a tree learner, it may not be improved by the use of Bookmaker weighting (Adaboost) or Kappa weighting (AdaKap) rather than a conventional uncorrected accuracy or error optimising learner. This raises an empirical question about whether boosting will work with different algorithms, and whether the form of chance-correction that corresponds to our analysis and hypotheses indeed performs best. Our second prediction is that failure of Adaboost and AdaKap and standard Adaboost with Rand Accuracy, can be expected at times, with “early stopping” due to the weak learner failing to satisfy the 1/2 condition. But for different distributions, and different weak learner optimization criteria, one can improve and another worsen.

Multiboost (Webb et al., 2011) seeks to avoid this “early stopping” by interleaving bagging amongst the Adaboost iterations – we use the Weka default “committee size” or interleave of 3 as in our experiments to test our second empirical question: Can Multiboost with Bookmaker Informedness (Multiboost) or Kappa (Multikappa) weighted accuracy achieve better boosting and overcome the hypothesized disadvantage of Multiboost due to the weak learner being optimized for Accuracy?

3 Data Sets and Algorithms

For comparability with the other work on multiclass boosting (e.g. Zhu et al. 2009), we use the same character set datasets as shown in Table 1 along with their number of classes, attributes and instances. 2x5-fold Cross Validation was used for all experiments. As we in general had 26 English letters, 26, 260 and 2600 boosting iterations were tested (a weak learner may boost just one class). Graphs for the Multiboost vs Adaboost comparisons show Standard Deviations (red extension bars) and 2 standard error Confidence Intervals (black whiskers). Table 1. Datasets and Sizes. Tra indicates Training set only used (for 2x5-CV).

We have explored the chance-corrected boosting of Naïve Bayes with results as summarized in Fig. 1. It is noted that only for one data set, Vowel, was significant boosting achieved, and for one, Letter, all the usual chance-corrected measures made things marginally worse (but not to a degree that is either practically or statistically significant). Also as expected, neither of the chance correction measures was particularly effective, and there was no clear advantage of Boosting over Kapping or vice-versa, except that on the one dataset where any boosting happened, Booking was faster than Kapping (with a difference that was only marginally significant at p<0.05 for 26 iterations, and disappeared completely by 260 iterations), but they did not do significantly differently from standard Adaboost with Rand Accuracy, which actually appeared to be best for this dataset, as well as for Letter as previously noted. Bayesian approaches were thus not pursued further.

[Figure 1. Boosting Naïve Bayes rarely works and chance-correction makes little difference. 2x5-CV (+260 iterations) and we show better results in Figure 3.]

Archived at Flinders University: dspace.flinders.edu.au
4 Results

For weak and strong rule/tree learners where boosting was expected and confirmed (Weka’s REPTree and Simple Cart), we tested MultiBoost as well as AdaBoost using Accuracy, Cohen Kappa and Powers Informedness, and selected results are shown in Figs 2 & 3. For reference both the single level and complete Decision Tree are shown as baseline (REPTree Decision Stump) and baseline (REPTree Full Tree). Table 2 also shows 260 cycles for a weak base learner and 26 for 2 strong base learners for additional datasets with SimpleCART as the additional strong base learner.

Although we have tested the family of Perceptron and SVM learners extensively, they are beyond the scope and space of this paper, where for fairness we concentrate on tree-type weak learners as originally proposed for AdaBoost, but we note that similar results pertain. We cannot confirm that Multiboost is better than AdaBoost but rather they seem evenly balanced as to which is best: we see no evidence of avoiding early surrender, but suggest that the bagging mechanism is a matter for future work, but we see slightly better performance on stronger base learners.

For Decision Stumps (DS in Figs & Table 2) there are two character recognition cases where Multiboost was significantly better (Vowel and Isolated) and for the other datasets AdaBoost seems to be a bit better. In all cases, the uncorrected accuracy versions failed to boost, but boosting was achieved with corrected accuracies. In two of the six cases (Opt and Hand), AdaBoost was already comparable with or better than the full REPTree learner, and Multiboost and AdaKap performed slightly less spectacularly. It is telling that standard AdaBoost is uncompetitive, and that even with chance-corrected boosting, it mostly fails to attain the REPTree baseline. In Fig. 2 for both experiments we use 26 and 260 iterations of DS boosting, but in Fig. 3 we show 2600 iterations of AdaBoost gives no further gain.

When boosting a stronger REPTree learner (noting that the Decision Stump learner is REPTree restricted to a single branch decision), the story is quite different: in all cases all boosting approaches achieved significant improvement over REPTree, with Multiboost apparent best in four of the six cases (similar results for all boosters were achieved for Pen and Opt, but as we approach 100% accuracy, there is less scope to show their merit, and these had the underlying learners with the highest inherent accuracy). The results for boosting SimpleCART are very similar, and often slightly better than for REPTree as seen in Table 2.

5 Conclusions

We have extended chance-corrected adaptive boosting of standard weak learners to include bagging iterations according to the MultiBoost algorithm. Chance correction is found to make a considerable difference to the performance of both AdaBoost.M1 and MultiBoost (with three iterations of AdaBoost.M1 to one of Bagging). Indeed, for a weak learner it tends to make the difference between boosting nicely, and not boosting at all, whilst for a stronger learner, better results tend to be achieved, and no worse results were achieved, except for two of the additional datasets shown in Table 2 where for Sick neither REPTree nor Simple CART showed improvement either, and for Hypothyroid the boosting with Accuracy failed and both Kappa and Informedness regressed the strong learners minor improvement above baseline.

Compared with other variants of boosting or AdaBoost, no inbuilt learning or regression mechanism is required, and no probability or plausibility or confidence rating or ranking needs to be generated for the weak learner: a standard learner can be used and no extension is required. However, it is usually better to start with a strong learner.

Moreover, it is not necessary to run separate training sessions for each class – learning across all classes simultaneously is possible for base classifiers that support this.

On the other hand, boosting performance for Naïve Bayes was spectacularly absent, with only one dataset achieving boosting, and no chance-correction mechanisms showing any advantage versus accuracy. The Naïve Bayes learner is significantly different from a Tree Learners, and this apparent independ- ence may make it suitable for use in multi-classifier variants of boosting, bagging or stacking, based on ensemble fusion techniques involving variation to the classifier rather than just the selection or weighting of data, and using optimization of weights. A major deficiency of this work is that we used only base learners that were optimized in terms of uncorrected accuracy or error, and it is noted (Powers, 2011) that such optimization can actually make things worse in chance-correct or cost-penalty terms. There is thus a strong chance that the weak learner will detrain and thus not satisfy the boosting condition, and this is particularly likely for Informedness, but less likely for Kappa which is more closely related to Accuracy, and tends more to move with Accuracy, although its divergence from Informedness is itself a source of reduced performance. This explains why often Kappa will seem to do better than Informedness, which should do better on theoretical grounds given a chance-correct weak learner.
Table 2: AdaBoost & MultiBoost by Accuracy, Kappa and Informedness, vs Decision Stump, REPTree & SimpleCART. Wins within 5% equivalence range are counted; plus Boosts & Losses outside 5% equivalence range are weak learner. 
SimpleCART is not shown for space reasons, but is generally slightly better than RT and slightly worse than Boosted SC. 
Informedness is shown but if there is a sig. qualitative/directional discrepancy with Accuracy or F1 this is marked A|F.

<table>
<thead>
<tr>
<th>Informedness</th>
<th>AdaBoost.1</th>
<th>Baseline</th>
<th>TreeLine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset</td>
<td>AccRT26B</td>
<td>AccD26B</td>
<td>AccRT26B</td>
</tr>
<tr>
<td>handwritten</td>
<td>0.84</td>
<td>0.93</td>
<td>0.84</td>
</tr>
<tr>
<td>hyperthyroid</td>
<td>0.89</td>
<td>0.92</td>
<td>0.88</td>
</tr>
<tr>
<td>iris</td>
<td>0.93</td>
<td>0.93</td>
<td>0.93</td>
</tr>
<tr>
<td>islet</td>
<td>0.92</td>
<td>0.92</td>
<td>0.92</td>
</tr>
<tr>
<td>letter</td>
<td>0.94</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>nursery</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>optdigits</td>
<td>0.96</td>
<td>0.97</td>
<td>0.96</td>
</tr>
<tr>
<td>pendigits</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>segment</td>
<td>0.96</td>
<td>0.97</td>
<td>0.97</td>
</tr>
<tr>
<td>sick</td>
<td>0.92 AF</td>
<td>0.92</td>
<td>0.92</td>
</tr>
<tr>
<td>waveform</td>
<td>0.73</td>
<td>0.72</td>
<td>0.72</td>
</tr>
<tr>
<td>Average</td>
<td>0.91</td>
<td>0.92</td>
<td>0.92</td>
</tr>
<tr>
<td>SigWins</td>
<td>8</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>SigLosses</td>
<td>10</td>
<td>3</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>AccRT26B</th>
<th>AccD26B</th>
<th>AccRT26B</th>
<th>AccD26B</th>
</tr>
</thead>
<tbody>
<tr>
<td>handwritten</td>
<td>0.86</td>
<td>0.89</td>
<td>0.84</td>
<td>0.93</td>
</tr>
<tr>
<td>hyperthyroid</td>
<td>0.98 AF</td>
<td>0.99</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>iris</td>
<td>0.94</td>
<td>0.91</td>
<td>0.93</td>
<td>0.93</td>
</tr>
<tr>
<td>islet</td>
<td>0.92</td>
<td>0.92</td>
<td>0.92</td>
<td>0.92</td>
</tr>
<tr>
<td>letter</td>
<td>0.94</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
</tr>
<tr>
<td>nursery</td>
<td>0.99</td>
<td>1.00</td>
<td>0.99</td>
<td>1.00</td>
</tr>
<tr>
<td>optdigits</td>
<td>0.96</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
</tr>
<tr>
<td>pendigits</td>
<td>0.98</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>segment</td>
<td>0.97</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>sick</td>
<td>0.86</td>
<td>0.86</td>
<td>0.83</td>
<td>0.86</td>
</tr>
<tr>
<td>waveform</td>
<td>0.85</td>
<td>0.93</td>
<td>0.89</td>
<td>0.94</td>
</tr>
<tr>
<td>Average</td>
<td>0.92</td>
<td>0.92</td>
<td>0.92</td>
<td>0.92</td>
</tr>
<tr>
<td>SigWins</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>SigLosses</td>
<td>8</td>
<td>8</td>
<td>3</td>
<td>9</td>
</tr>
</tbody>
</table>

Figure 3: 2x5CVs, 26 iteration AdaBoost & MultiBoost with & without chance correction on REPTrees. 2x5CV 26, 260 & 2600 iteration AdaBoost with Informedness is shown for comparison (overlap with Fig. 2). Standard deviation/baseline range (Decision Stump) and treeline range (REPTree) are also shown for reference.
6 Future Work

Chance correction has been advocated for decades, but only recently has being incorporated into learners, starting with boosting. It is clear that it should be incorporated into base learners as well, and further studies are needed to explore those existing learners that optimize Informedness or some other chance-corrected measure.

In addition further variations and combinations on boosting, bagging and stacking would seem to be worth exploring to address the limitations of particular weak learners and ensure that boosting is allowed to continue. In particular, techniques that revert to lower K-learners on failure of the weak learner, would gain the best of both worlds – fast multiclass learning where possible, and solid but slow low cardinality or single class learning when not. As noted above, this includes exploiting the sensitivity of Multiboost and its chance-corrected variants to the number of bagging and boosting iterations.

We identify the fact that weak learners are still optimizing an uncorrected measure as the major obstacle to achieving the theoretical performance of chance-corrected boosting, and in Table 2 we have not with A resp. F cases where the Accuracy res. F1 have risen but chance-corrected measures fell. We are working on general modifications/.wrappers for broad classes of learner to address this issue, including a specific focus on ANNs, SVMs and Decision Trees.

It is also a priority to explore boosting of learners that are sensitive to noise and don’t have the convexity constraints of AdaBoost, including learners that are based on switching and can explore and use informative noise signals. Since boosting works well with tree learners, such a tree-like approach would produce a consistent but potentially more comprehensible model due to the structural risk minimization properties of boosting and the noise sensitivity minimization properties of switched boosting. However we are also exploring performance with SVM and MLP learners with projective inconsistency.

Our focus here was the language/character multiclass problems, but we also have more general robotic vision, and other applications. However, the diverse natures of these problems, as illustrated by the other half dozen datasets in Table 2, remain to be characterized and understood. It is particularly important to explore what difference chance-correction makes in practical applications, and an obvious application where AdaBoost is a mainstay component, is face finding and object tracking (Viola & Jones, 2001).

This paper has concentrated on two particular kinds of ensemble technique: boosting and bagging in combination with boosting. One of the explanations of why these techniques work, and why boosting is in general more effective than bagging, is that the different subsets of instances that are selected for learning, and thus the different trained weak learners, correspond to different weightings on the features as well as the examples. Techniques like that explore feature evaluation and selection, including ensemble techniques like Random Forests and Fenting, more directly select features. When features have different sources (e.g. biomedical sensors, audio sensors and video sensors combined) or have spatiotemporal interrelationships (e.g. pixels or MRI voxels or EEG electrodes sampled at a specific rate), then there is additional structure that may be usefully explored.

AdaBoost, AdaBook and AdaKap may all be used reasonably effectively as early fusion techniques because of these implicit feature selection properties, and in our current work the chance-correction advantage is again clear, although this is beyond the scope of this paper. Nonetheless there seems to be a lot more room for improvement including selecting features and combining weak classifiers in ways that bias towards independence (or decorrelation) rather than using simple majority or convexity fusion techniques as implemented in traditional boosting. This is something else we are exploring.

We have also glossed over the existence of a great many other boosting algorithms, and the known limitations of convex learners such as AdaBoost is in dealing with noise. These convex learners are Perceptron-like and the AdaBoost learner is a simple linear combination of the trained weak learners, and are known not to be able to handle label noise. The original boosting algorithm was not robust to noise, and no boosting or voting ideas, and further work is needed on variants of boosting that don’t overtrain to noise like AdaBoost can, but are insensitive to the occasional bias introduced by label noise, or the regular variance introduced by attribute and measurement noise, or in the kind of artefacts and punctuated noise we get in signal processing, including EEG processing, audio speech recognition, and video image or face tracking. The idea is that successive stages bump an instance up or down in likelihood but our mislabelled instance is not repeatedly trained with increasing weight until it is labelled “correctly” (Long & Servidio, 2005, 2010).

We advocate the use of chance-corrected evaluation in all circumstances, and it is important to modify all learning algorithms to use a better costing. Uncorrected measures are deprecated and should never be used to compare across datasets with different prevalences or algorithms with different biases.

ACKNOWLEDGEMENTS

This work was supported in part by the Chinese Natural Science Foundation under Grant No. 61070117, and the Beijing Natural Science Foundation under Grant No. 4122004, the Australian Research Council under ARC Thinking Systems Grant No. TD0669874, as well as the Importation and Development of High-Caliber Talents Project of Beijing Municipal Institutions.

REFERENCES


Archived at Flinders University: dspace.flinders.edu.au
Maximizing common evaluation measures such as Accuracy, Precision, Recall and F1, or minimizing most commonly employed error measures, absolute or relative, does not correctly optimize your problem and is not comparable across different datasets, different contexts, different algorithms or even different parameterizations, where these differences lead to different prevalences or biases. Prevalence of a class is how often it occurs in the real world, while the Bias of a label is how often you predict it. Obviously for a perfect solution, these should match, but sometimes some cases are worth more than others, or are more difficult than each other.

\section*{Many approaches have been proposed over the last century to address the problem of bias by correcting measures for chance, or by devising measures that take into account the varying costs and prevalences. This is different from just looking for evidence that the results are significantly different from chance, and ideally they are directionally rather than just indicating a difference. Some common alternatives are the various forms of Kappa, and the various forms of trade off of two traditional statistics, of which ROC (Receiver Operating Characteristics) is a relatively sound approach, while plots or averages of Recall vs Precision are very unsound methods. Kappa involves making an estimate of the expected performance due to chance, and then renormalizing this based on available room for improvement over chance: Kappa = (1 + Prior) / (1 + Chance). Kappa has been derived and reinvented many times under different names [1,2], and represents the cost placed on wins and losses by a Bookmaker, or the same marking system as is used for cancelling out the effect of guessing in multiple choice tests. It is also closely related to Correlation, and Kappa, Informedness and Correlation are well defined for the multiclass case, unlike Recall, Precision and F1 which consider only a single class! Binary Informedness can be expressed in ROC terms, tpr–fpr, or as a form of Kappa: \( \text{Informedness} = \frac{1}{\text{ROC}} \) which is a well defined probabilistic interpretation as the probability of making an informed decision (rather than guessing) [1].

\section*{Boosting, Bagging & Classifying}

If uncorrected evaluation is so bad, and chance-corrected evaluation so important, doesn’t this mean our learning algorithms should be optimizing chance-corrected measures? Yes, indeed! Some algorithms (like ANNs and SVMs) have a bias to match Prevalence and Bias (when all measures become equivalent), whilst others (like Decision Trees and AdaBoost) tend to end up making an assumption that amounts to assuming equal numbers of positives and negatives (np = pp or fp=fn). Acc\(_{\text{acc}}\) = \( \frac{1}{\text{K}} \) and Err\(_{\text{acc}}\) = \( \frac{1}{\text{K}} \) are simple formulae that will map any 0\%–100\% Kappa-like chance-corrected measure to a 0%–100% accuracy or Error-like measure than can be plugged straight into boosting algorithms like AdaBoost.

Boosting works by looking at the cases an initial Base Classifier gets wrong, and working harder on those – theoretically it can work as long as the Base Classifier always does better than chance. AdaBoost uses the odds ratio, accuracy to error, to do the reweighting. If we plug in Cohen Kappa, we call it AdaKap, and if we plug in Bookmaker Informedness, we call it AdaBook. We can also plug these into MultiBoost, to extend AdaBoost by introducing cycles of Bagging – randomly selected ‘bags’. How well AdaBoost, AdaBook and AdaKap will do depends on the learner and what it optimizes. For example none of them work well with Naïve Bayes (Fig. 1 in the paper).

AdaBoost.M1 is a version of AdaBoost that turns K-class problems into one-versus-all 2-class problems, but tends not to work well as it requires the Base Classifier to achieve an accuracy of at least 1/2 whereas chance is likely to be more like 1/4. But AdaBoost and AdaKap don’t have this problem so will tend to do much better, although sometimes a weak Base Learner will still optimize an incorrect measure and stop the boosting prematurely ‘early surrender’ 1. Where a strong Base Learner is used this is unlikely to occur, so AdaBoost and AdaKap are only marginally ahead. But as Kappa is closely related to Accuracy it often does better than the theoretically optimal Bookmaker version.

Figures 2 and 3 illustrate all these variants of AdaBoost with a weak Decision Stump, an easier learner (splits with one attribute to make its decision) as well as with a strong REPTree learner (repeats with other attributes to grow the tree further).

\section*{References}


