Comparing Knowledge-Based Sampling to Boosting

Martin Scholz

University of Dortmund, 44221 Dortmund, Germany,

scholz@ls8.cs.uni-dortmund.de,

http://www-ai.cs.uni-dortmund.de/

July 25, 2005

Abstract

Boosting algorithms for classification are based on altering the initial distribution assumed to underly a given example set. The idea of knowledge-based sampling (KBS) is to sample out prior knowledge and previously discovered patterns to achieve that subsequently applied data mining algorithms automatically focus on novel patterns without any need to adjust the base algorithm. This sampling strategy anticipates a user's expectation based on a set of constraints how to adjust the distribution. In the classified case KBS is similar to boosting. This article shows that a specific, very simple KBS algorithm is able to boost weak base classifiers. It discusses differences to AdaBoost.M1 and LogitBoost, and it compares performances of these algorithms empirically in terms of predictive accuracy, the area under the ROC curve measure, and squared error.

1 Introduction

The task of finding novel patterns in the presence of prior knowledge has recently been addressed by a technique referred to as knowledge-based sampling [1, 2]. For a specific rule induction task a simple reweighting strategy has been shown to produce diverse rulesets, each rule describing a separate, independent "subgroup" of its own. In [3] some early successful experiments with the KDD cup data of 2004 are reported, based on using the same reweighting strategy to boost decision trees. This article compares the corresponding algorithm to the best known and practically most relevant boosting algorithm AdaBoost.M1, and more briefly to the regression-based LogitBoost. Both boosting algorithms are described in section 2, before section 3 repeats the main ideas guiding knowledge-based sampling. A subsequent discussion compares the sampling technique to boosting from a methodological point of view, and it explains how this kind of sampling can be used to boost weak base classifiers. The claims are empirically evaluated in section 4. The crisp predictions of AdaBoost.M1 are compared to KBS in terms of predictive accuracy. The proposed KBS boosting algorithm is designed to yield soft predictions, which allows for a finer-grained comparison to LogitBoost. The rankings induced by the KBS-boosting and LogitBoost are compared in terms of AUC. To evaluate the absolute predictions the squared error is applied as a second measure. Section 5 summarises the results shown in this report.

2 Boosting Weak Classifiers

The ensemble method of boosting allows to induce a set (or an ensemble) of classifiers by repeatedly running a "weak" base learner. In each iteration the training set is slightly altered, assigning higher weights to subsets that have been misclassified in previous iterations. This technique allows to improve on the classification performance of the weak learner, and to reach arbitrarily accurate combined classifiers, as long as each weak hypothesis is slightly better than random guessing [4]. The first boosting algorithm of practical interest was AdaBoost.M1 [5], depicted in figure 1. It combines single classifiers by a weighted majority vote, deriving the weights based on individual accuracies. As discussed in [5] the voting strategy is identical to the Bayes optimal decision rule if the conditional independence assumptions of NaïveBayes [6] applies to the predictions of base classifiers. AdaBoost.M1 has been analysed elaborately. It has been shown to maximise the margin [7], and it can also be derived in terms of projecting entropy [8]. As an important property for practical applications it has been reported to hardly overfit to the training data.

1. Let D_1 denote the uniform distribution over training set $\mathcal{E} = \langle x_1, y_1 \rangle, \ldots, \langle x_m, y_m \rangle$.
2. For $i = 1$ to n do
   (a) Call WEAKLEARNER(D_i, E), find accurate model $h_i : \mathcal{X} \rightarrow \mathcal{Y}$.
   (b) $\epsilon_i := Pr_{\langle x,y \rangle \sim D_i} [h_i(x) \neq y]$ (weighted empirical training error)
   (c) $\beta_i := \frac{\epsilon_i}{1-\epsilon_i}$
   (d) $D_{i+1}(x_j, y_j) := \frac{D_i(x_j,y_j)}{Z_i} \cdot \begin{cases} \beta_i, \text{ if } h_i(x_j) = y_j \\ 1 \text{ , otherwise} \end{cases}$
   with Z_i being a normalisation factor.
3. Output the final hypothesis:

$$h(x) := \arg\max_{y \in \mathcal{Y}} (\sum_{i:h_i(x)=y} \log \frac{1}{\beta_i})$$

Figure 1: Algorithm ADABOOST.M1

1. Let D_1 denote the uniform distribution over training set $\mathcal{E} = \langle x_1, y_1 \rangle, \ldots, \langle x_m, y_m \rangle$.
2. For all $\langle x_i, y_i \rangle \in \mathcal{E}$ initialise $p_i = 1/2$ (estimator of $Pr\,[y_+ \mid x_i]$) and $F_0(x_i) := 0$.
3. For $j = 1$ to n (and $i = 1$ to m) do
   (a) Update working responses: $z_i \leftarrow \frac{y_i - p_i}{p_i \cdot (1-p_i)}$
   (b) Update weights: $w_i \leftarrow p_i \cdot (1 - p_i)$
   (c) Select a function $f(x)$ predicting z based on least squares regression.
   (d) Let $F_j(x) := F_{j-1}(x) + \frac{1}{2} f(x)$
   (e) and $\widehat{Pr}_j(y_+ \mid x) := e^{F_j(x_i)} \left(e^{F_j(x_i)} + e^{-F_j(x_i)} \right)^{-1}$
   (f) For $\langle x_i, y_i \rangle \in \mathcal{E}$ update: $p_i \leftarrow \widehat{Pr}(y_+ \mid x_i)$
4. Output function $\widehat{Pr}_j : \mathcal{X} \rightarrow [0, 1]$ estimating $Pr\,[y_+ \mid x]$.

Figure 2: Algorithm 2-class LogitBoost

Despite some minor short-comings the original version proposed in [5] is probably still the most popular boosting algorithm in practice. Top-down induction of decision trees is known to be a well suited base learner. As some authors report, for many datasets boosting a very restricted class of decision trees performs equally well or better, for instance the so called DecisionStumps, consisting of a single node, only.

Friedman pointed out, that AdaBoost.M1 is very similar to a technique known as *additive logistic regression* in statistics [9]. This connection inspired some new boosting algorithms, the probably best known of which is LogitBoost, depicted in figure 2. In contrast to AdaBoost.M1 this boosting procedure allows to estimate class probabilities. It is often used in combination with DecisionStumps or similar simple base learners.

3 Knowledge-based Sampling

This section introduces a new boosting algorithm based on a constraint-guided example reweighting or sampling approach. Knowledge-based sampling was originally suggested to incorporate prior knowledge into supervised data mining algorithms by means of sampling [1, 2]. It addresses the question of how to pre-process the data in the presence of prior knowledge, so that subsequent data mining steps do not report previously found patterns again, neither explicitly, nor patterns describing similar instances, but rather focus on uncorrelated new patterns. The sampling strategy suggested in [1] succeeds if the task is to find novel or surprising patterns, and if novelty is formalised by deviation from a user's expectation. The "expectation" is required to be given in a form that allows to compute probability estimates for the target attribute. A task referred to as *subgroup discovery* [10, 11] aims at finding interesting rules, where interestingness is formalised in terms of a utility function. This function combines deviation from expectation and coverage.

Patterns that are found in one iteration extend the user's expectation or prior knowledge in the next one. Hence, in each iteration the sampling procedure produces training sets that are "orthogonal" to the combined probability estimate corresponding to the prior knowledge. This aspect is very close to boosting classifiers.

3.1 Constraint-guided sampling

Before going into detail the idea of removing prior knowledge by biased sampling is formulated in terms of constraints. Formally, this step defines a new distribution, as close to the original function as possible, but orthogonal to the estimates produced by available prior knowledge. Monte Carlo methods like rejection sampling allow to sample with respect to the new distribution, provided with just a procedure to sample from the original distribution [12]. As a result of switching distributions the utility functions – when applied to these kind of samples – are "blinded" regarding the parts of the data that can be concluded from prior knowledge. All that is accounted for is the unexpected component of each rule or model.

For the experiments described in this article the only kind of "prior" knowledge are the weak models yielded by preceding iterations. In principle the techniques allow for an incorporation of "real" prior knowledge into boosting, an idea that has also been discussed in [13]. Manually encoded prior knowledge can be expected to show a similar behaviour as those rules that are typically found in early iterations. Unfortunately it is hard to elicit prior knowledge for the domains relevant to public datasets in a fashion that can be expected to be representative for practical applications. For this reason this aspect is not evaluated empirically.

For a given instance space $\mathcal{X}$ and nominal target attribute $\mathcal{Y}$ examples are expected to be sampled i.i.d. from an initial distribution $D : \mathcal{X} \to \mathbb{R}^+$. Each example $x \in \mathcal{X}$ is assigned a class by an unknown but fixed function $C : \mathcal{X} \to \mathcal{Y}$. For $k \in \mathbb{N}$ let $h : \mathcal{X} \to \mathcal{Y}$ denote a weak hypothesis from hypothesis space $\mathcal{H}$, predicting a value of the target attribute.

As a first constraint the new distribution D' to be constructed should no longer support the knowledge encoded in the hypothesis h, so the predictions $h(x)$ and the true label should be independent considering D':

$$(\forall y, y' \in \mathcal{Y}) : Pr_{x \sim D'} [C(x) = y \mid h(x) = y'] = Pr_{x \sim D'} [C(x) = y] \tag{1}$$

It is easy to see, that in all other cases the same hypothesis allows to derive further information about the true label's conditional distribution given the prediction of h.

As a second constraint the probability to observe an example that belongs to a specific class $y \in \mathcal{Y}$, or is predicted to have a specific class $y' \in \mathcal{Y}$ should not change from D to D', since it is sufficient and possible to remove only the correlation between the random variables $\mathcal{Y}$ and $h(\mathcal{X})$:

$$(\forall y \in \mathcal{Y}) : Pr_{x \sim D'} [h(x) = y] = Pr_{x \sim D} [h(x) = y] \tag{2}$$

$$(\forall y \in \mathcal{Y}) : Pr_{D'} [C(x) = y] = Pr_{D} [C(x) = y] \tag{3}$$

The former of these equalities makes sure that the class skew does not change, which would result in an implicitly altered cost model for misclassifying examples. The latter equality expresses that e.g. rules should apply equally often for both distributions, avoiding to introduce unnecessary skews to the marginal distribution.

Finally, within each partition sharing the same predicted and true class the new distribution is defined proportionally to the initial one. The simple reason is that having just a hypothesis h as prior knowledge all instances within one partition are indistinguishable. A change to the conditional probabilities within one partition means to prefer some instance over the others, despite their equivalence with respect to the available prior knowledge. This translates into the following constraint:

$$
\begin{aligned}
&(\forall x \in \mathcal{X})(\forall y, y' \in \mathcal{Y}): \\
&\quad Pr_{x\sim D'}(x \mid h(x) = y', C(x) = y) = Pr_{x\sim D}(x \mid h(x) = y', C(x) = y)
\end{aligned}
\tag{4}
$$

Constraints (1)-(4) induce a unique target distribution. The following definition eases notation.

Definition 1 *The* LIFT of an example $x \in \mathcal{X}$ *for a given class* $y \in \mathcal{Y}$ *and hypothesis* $h : \mathcal{X} \to \mathcal{Y}$ *is defined as*

$$\text{LIFT}_D(x \to y \mid h) := \frac{Pr_{x\sim D}\left[h(x) = y', C(x) = y\right]}{Pr_{x\sim D}\left[h(x) = y'\right] \cdot Pr_{x\sim D}\left[C(x) = y\right]}$$

Theorem 1 *For any initial distribution* D *and partitioning ruleset* $h \in \mathcal{H}$ *the constraints (1)-(4) are equivalent to*

$$Pr_{D'}(\langle x, y\rangle) = Pr_D(\langle x, y\rangle) \cdot \left(\text{LIFT}_D(x \to y \mid h)\right)^{-1}.$$

A proof is given in [1]. Theorem 1 defines a new distribution to sample from, given a hypothesis h as prior knowledge. Assuming a single hypothesis is not very restrictive, since it is possible to directly incorporate each new weak hypothesis into a single "global" model. A simple strategy based on this idea is evaluated in the remaining article: The theorem is applied iteratively. Hypothesis h_i is selected based on distribution D_i. Distribution D_{i+1} is defined by applying theorem 1 to D_i and h_i. The inverse of this strategy allows to approximately reconstruct the original distribution D_1 as a combination of the single hypotheses.

The following equalities illustrate how to compute estimates of the conditional distribution for a boolean target attribute $\mathcal{Y} = \{y_+, y_-\}$. If $\mathcal{Y}$ is boolean then this estimate can be simplified by considering the odds, since

$$\beta(x) := \frac{Pr(x \to y_+)}{Pr(x \to y_-)}, \quad Pr(x \to y_+) = \frac{\beta(x)}{1 + \beta(x)}, \quad Pr(x \to y_-) = \frac{1}{1 + \beta(x)}.$$

1. Let D_1 denote the uniform distribution over $\mathcal{E} = \langle x_1, y_1 \rangle, \ldots, \langle x_m, y_m \rangle$.
2. For $i = 1$ to n do
   (a) Call WEAKLEARNER(D_i, E) to find an accurate model $h_i : \mathcal{X} \rightarrow \mathcal{Y}$.
   (b) Compute $\text{LIFT}_{D_i}(x \rightarrow y \mid h_i)$ applying definition 1.
   (c) Let $D_{i+1}(x_j, y_j) := D_i(x_j, y_j) \cdot (\text{LIFT}_{D_i}(x_j \rightarrow y_j \mid h_i))^{-1}$.
3. Output $\{h_1, \ldots, h_n\}$ and the LIFT values. Predict $Pr(y \mid x)$ with eqn. (5).

Figure 3: Algorithm KBS

The term $\beta(x)$ can then be estimated based on $\{h_1, \ldots, h_n\}$, a sequence of n hypothesis, each of which is the result of a separate iteration:

$$\beta(x) \approx \frac{Pr(y_+)}{Pr(y_-)} \cdot \prod_{i=1}^{n} \frac{\text{LIFT}_{D_i}(x \rightarrow y_+ \mid h_i)}{\text{LIFT}_{D_i}(x \rightarrow y_- \mid h_i)} \tag{5}$$

Please note the subscripts D_i of the LIFT terms.

Figure 3 shows the knowledge-based sampling algorithm, which is analysed on the following pages. To the best of the author's knowledge this is the most simple boosting algorithm proposed in the literature so far. Eqn. (5) is used by the algorithm to operationalise the reconstruction of D_1. It is discussed in more detail in the next subsection. For classification the most probable class is computed from the estimated conditional distribution.

3.2 How KBS boosts base classifiers

The KBS algorithm consists of a strategy for reweighting examples, and one for combining induced base classifiers. This subsection discusses the underlying idea of these strategies and explains why they match.

Let D_1 denote the original distribution the learner tries to approximate. Whenever the base learner is run for a distribution D_i a weak hypotheses is selected that captures correlations between $\mathcal{X}$ and $\mathcal{Y}$. These correlations are then "removed" in the step from D_i to D_{i+1}. With a growing number of iterations the conditional distributions of $\mathcal{Y}$ given $\mathcal{X}$ converge towards the class priors.

To understand how the resulting model is applied, the definition of an *atomic subset* of $\mathcal{X}$ is useful. An atomic subset is a subset having the property that the vector of all n predictions $\langle h_1(x), \ldots, h_n(x)\rangle$ is identical for all contained examples x. Assuming that the conditional distribution is approximately equal to the priors for D_{n+1} it is possible to stepwise "retract" the application of theorem 1 used to reach this distribution. If the probability of seeing an example with a specific label under D_{n+1} is equivalent to the class prior, then the distribution of classes under D_n can be derived by *multiplying* the priors of the atomic subset with the corresponding LIFT values for each class and prediction. Renormalising and iteratively applying the corresponding LIFTs for each h_i the initial D_1 can be reconstructed precisely for this subset.

Of course in practice one cannot assume to end up with an "ideal" D_{n+1} as assumed, but the idea still applies unless strong correlations between $\mathcal{X}$ and $\mathcal{Y}$ are not captured by the weak hypotheses. Applying just a single base classifier, the conditional distribution of classes given its prediction constitutes the probability estimator. Even this most coarse approximation is not unintuitive from a practical point of view.

The classification model is not altered when considering the logarithm of eqn. (5) instead of the original estimate. In this form it can be seen that the KBS strategy of constructing a decision rule is close to additive modelling as described in [9]. This is due to the fact that the LIFT ratios for h_i are estimated based on D_i, containing the remaining patterns (or residuals), only.

3.3 Learning rate

The learning rate is analysed for two-class problems only, and for the simplified situation in which the boosting procedure yields a set of equally accurate base classifiers with conditionally independent predictions. This does not clarify all questions, but allows to compare the learning rates of KBS and ADABOOST.M1 qualitatively.

Let n denote the number of iterations, $\text{ACC} > 1/2$ be the common accuracy of all weak hypotheses, and $\gamma = \text{ACC} - 1/2$ be the advantage over random guessing. Given that the predictions of all base learners are conditionally independent the (expected) accuracies with and without reweighting are equivalent. In [5] this specific case has been analysed for AdaBoost.M1: The rule used by AdaBoost.M1 to combine models assigns equal weight to each of the base classifiers, because of the common accuracy, and regardless of which are the predicted classes. This implies that the most frequently predicted class gets selected. The expected error rate ϵ is thus bounded by the probability of at least half of the weak models voting for the wrong class. Translated to more technical terms this is equivalent to seeing the less probable class in a Binomial experiment at least $k = n/2$ times, which is bounded by (according to Chernoff):

$$\begin{aligned}
\epsilon &\leq \left(\frac{n - n\cdot \textsc{Acc}}{n-k}\right)^{n-k}\left(\frac{n\cdot\textsc{Acc}}{k}\right)^{k} = \left(\frac{n(1-\textsc{Acc})}{n-n/2}\right)^{n-n/2}\left(\frac{n\cdot\textsc{Acc}}{n/2}\right)^{n/2} \\
&= (2(1-\textsc{Acc})\cdot 2\textsc{Acc})^{n/2} = ((1-2\gamma)(1+2\gamma))^{n/2} = \left(\sqrt{1-4\gamma^2}\right)^{n} \\
&\leq e^{-2n\gamma^2}
\end{aligned}$$

It can easily be seen that the same bound holds for KBS: Because of the assumed conditional independence the base classifiers may be combined applying NaïveBayes, which yields exact probabilities in this case and leads to the optimal Bayes' decision rule. The subscripts D_i of eqn. (5) may be ignored, because the Lift ratios are equivalent for all distributions. Without these subscripts eqn. (5) is just a reformulation of NaïveBayes[1]. This implies that KBS computes ratios and probabilities *precisely*, and succeeds no worse in selecting the most probable class than AdaBoost.M1.

Finally it is interesting to understand, why the samples produced by KBS make base classifiers rank models according to their contribution to the overall accuracy. To this end, please note that after KBS "samples out" a model h the accuracy of all overlapping (correlated) models with respect to the new distribution is reduced according to the degree of overlap. Because of constraint (1) the Lift of the subset with a common prediction $h(x)$ is 1. Accuracy can be rewritten as

$$\textsc{Acc}(h) = \sum_{y\in\mathcal{Y}} Pr(h(x)=y)Pr(C(x)=y)\textsc{Lift}((h(x)=y)\rightarrow(C(x)=y)),$$

which is linear in the Lift of covered subsets. This implies that using the reweighted examples the base classifier will favour models according to their

[1]Details on how to rewrite NaïveBayes accordingly are discussed in more detail in [1, 2], but are rather trivial.

independent contributions. For pairs of models that are correlated with respect to D_1 the LIFTs multiply, which improves on the convergence rate shown above. This reflects the fact that the model which has been trained later uses a LIFT estimate based on a sample in which the knowledge represented by the former model has already been anticipated.

3.4 Differences between KBS and boosting

In the following paragraphs ADABOOST.M1 and LOGITBOOST are compared to the KBS algorithm depicted in figure 3. ADABOOST.M1 and KBS both apply a ("weak") base learner several times to reweighted versions of the same example set $\mathcal{E}$. The main difference during training is the way examples are reweighted from one iteration to the next one. The reweighting scheme of KBS is based on theorem 1, so it meets all the constraints introduced in subsection 3.1.

The reweighting scheme of ADABOOST.M1 meets constraint (4), since all examples that are indistinguishable in terms of predicted and true class are rescaled by the same factor of $\frac{1}{Z_i}$ or $\frac{\beta}{Z_i}$. According to this reweighting scheme only correctly classified examples receive a lower weight. If ϵ_i denotes the training error of the latest model then the weight is reduced by a factor of $\beta_i = \frac{\epsilon_i}{1-\epsilon_i}$. Normalisation reweights misclassified examples implicitly. It is easily seen that by the choice of β_i the correctly classified subset receives the same weight as the incorrectly classified subset.

In general the reweighting scheme of ADABOOST.M1 does not satisfy constraint (1). The precision of the hypothesis usually depends on the specific prediction. As a consequence, in an example set reweighted by ADABOOST.M1 the target variable is often *not* independent of the weak hypothesis. In other words ADABOOST.M1 wastes information extractable from hypotheses. This is illustrated by the following example. Let h_i denote a rule covering a small subset of $\mathcal{E}$, e.g. with 99% positives, while the fraction of positives in the huge non-covered part is close to the default probability of e.g. 50%, then ADABOOST.M1 is not able to distinguish between covered and uncovered parts. Both, the reweighting during learning and the prediction during application of the model will be based on the overall global error rate, which can easily be as large as 45% for typical rule induction scenarios. A consequence of using a single quality estimate per model (β_i) the reweighting scheme of ADABOOST.M1 does not meet constraints (2) and (3), which can easily be validated along the same example.

In fact, one way of concluding the correctness of KBS is that it is identical to AdaBoost.M1 if the class priors of a two-class problem are equal (so Lifts are proportional to precisions) and if the precisions for all predictions are identical. If the precisions are not identical than KBS exploits this fact by "splitting" the global model into local ones, and by using the more precise accuracies for the corresponding subsets. KBS can be thought of as applying the reweighting scheme of AdaBoost.M1 without any change to these subsets, just based on the local precisions in this case.

Classification rules with low coverage and high precision illustrate the main difference between the reweighting schemes of AdaBoost.M1 and KBS. KBS-reweighting is more expressive, since the Lift values introduce a different precision for each prediction. This helps to reduce the training error with fewer iterations, which will often result in lower generalisation error due to the Occam's razor principle.

Compared to LogitBoost the KBS procedure is much simpler. There is no separate working response, since the base learning problems are not formulated in terms of regression. This means that apart from resampling (or instance weighting) neither the data nor the base learning task is changed by the procedure. Please note, that LogitBoost makes use of a continuous working response *and* of example weighting. As a final difference KBS does not transform the odds by the logistic function. It directly relies on empirically observed frequencies to estimate probabilities.

Dataset	J48	AdaBoost/J48	KBS/J48
Contraceptive	51.66%	51.66%	**52.27** %
German	71.80%	69.92%	**73.41** %
Hepatitis	60.65%	63.23%	**65.16** %
Sat-Image	86.13%	**89.74** %	88.55%
Quantum Physics	65.62%	67.60%	**68.21** %
Homology Prediction	99.66%	**99.77** %	99.76%

Table 1: Predictive accuracies for different datasets

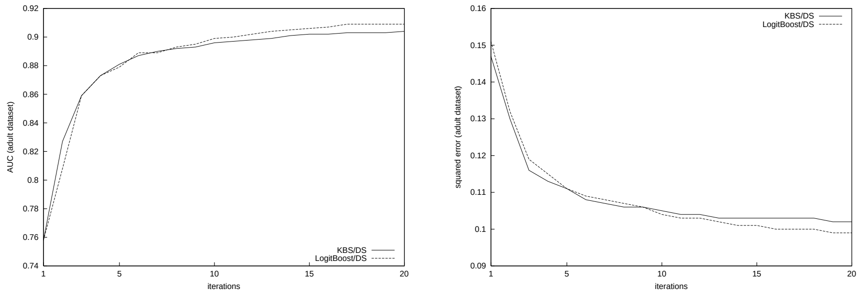

Figure 4: AUC and squared errors for KBS and LogitBoost using DecisionStumps

4 Experiments

KBS has been implemented as an operator of the learning environment YALE[2] [14] to evaluate the claims of the last sections experimentally. AdaBoost.M1, LogitBoost, DecisionStumps, and the decision tree induction algorithm J48 are part of the WEKA learning environment [15]. All datasets used in the following experiments are part of the UCI Machine Learning Library [16], except for the *Quantum Physics* and the *Homology Prediction* dataset which have been subject to the KDD cup 2004[3].

The first experiments compare the predictive accuracies achieved by the decision tree induction algorithm J48, AdaBoost.M1 on top of J48, and KBS on top of J48 for a few datasets. All accuracies reported in table 1 have been estimated by 10fold cross-validation. For the dataset *Contra-*

[2] http://yale.cs.uni-dortmund.de/

[3] http://kodiak.cs.cornell.edu/kddcup/

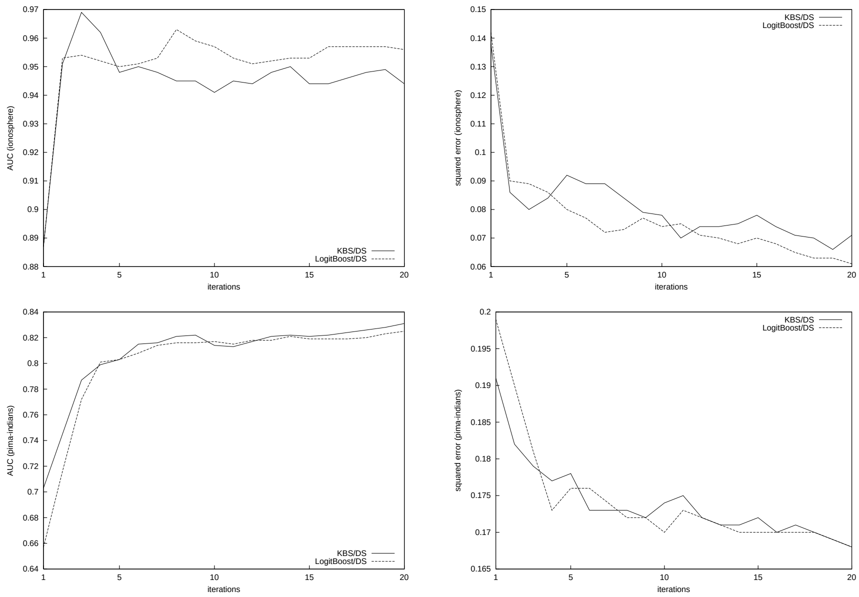

Figure 5: AUC and squared errors for KBS and LogitBoost using DecisionStumps

ceptive AdaBoost.M1 does not improve over the accuracy of J48. KBS manages to improve the accuracy if run for 20 iterations on 10% bootstraps of the data. All other performance values reported for AdaBoost.M1 and KBS are the results of 10 iterations without any internal bootstrapping. The results indicate that KBS is competitive to and often even outperforms AdaBoost.M1 using real-world datasets.

LogitBoost yields probability estimates, which may also be interpreted as confidence values for predictions. The area under the ROC curve (AUC) measure is used to compare the corresponding rankings for two-class classification problems, while absolute values of probability estimates are evaluated using the squared error criterion. The base learner applied in the subsequently presented experiments is DecisionStumps. In contrast to the experiments of the preceding subsection KBS is applied after a step of stratification as described in [1, 2]. This step can be considered to turn the base learner into one that optimises for a different utility function than accuracy, namely

$$\text{WRAcc}(h(x) \rightarrow y_+) := Pr(h(x)) \cdot (Pr(y_+ \mid h(x)) - Pr(y_+)) \, .$$

This function is commonly used for subgroup discovery [10, 17]. It allows to select even weaker models than commonly considered useful in the domain of boosting, because it is not necessary that the precision exceeds 50%. In combination with DecisionStumps stratification gave better experimental results.

Experiments have been carried out for three different datasets with boolean target attributes. The results shown in figures 4 and 5 have been estimated by 10fold cross-validation.

For the largest datasets *Adult* (30k examples) shown in figure 4 the AUC and squared error curves of both algorithms are very similar, but LogitBoost seems to converge a bit quicker. Overfitting is not an issue, so both AUC and squared error decrease monotonically. In contrast, for the smallest datasets considered (*Ionosphere*, 351 examples) KBS reaches the best AUC performance after 3 iterations, but then overfits (fig. 5). Overfitting can be addressed by cross-validation to find an optimal number of base learners in this case. LogitBoost performs slightly better regarding squared error on this dataset. For *Pima-Indians* (768 examples) KBS outperforms LogitBoost marginally.

5 Conclusion

This article compared the boosting algorithms AdaBoost.M1 and LogitBoost to the knowledge-based sampling algorithm recently introduced to find new interesting patterns in the presence of prior knowledge. It was pointed out that AdaBoost.M1 does not meet all the constraints that motivated the presented sampling technique. KBS uses a more expressive interpretation of the base learner, as it is able to distinguish between the precisions for different predictions. This helps to reduce the training error more quickly. It has been explained why KBS is capable of boosting (weak) base classifiers like decision trees and stumps. This claim has also been shown empirically, using real-world datasets. For 4 of 6 datasets KBS outperforms AdaBoost.M1. Surprisingly, according to further experiments the probability estimates of KBS are even competitive to the more complex LogitBoost, which uses example weights and continuous working responses in a regression framework. Keeping in mind the simplicity of the reweighting scheme of KBS the successful boosting capabilities deserve further attention. Future work will have to investigate more sophisticated knowledge-based algorithms in the scope of supervised Data Mining. A combination with data streams is a straightforward extension, since KBS seems to perform especially well when applied to independent samples in each iteration.

Acknowledgements

This work has been supported by the Collaborative Research Center "Reduction of Complexity for Multivariate Data Structures" (SFB 475) of the German Research Foundation (DFG).

References

[1] Scholz, M.: Knowledge-Based Sampling for Subgroup Discovery. In Morik, K., Boulicaut, J.F., Siebes, A., eds.: Local Pattern Detection. Volume 3539 of Lecture Notes in Computer Science. Springer (2005) 171–189

[2] Scholz, M.: Sampling-Based Sequential Subgroup Mining. In: Proc. of the 11th ACM SIGKDD Int. Conf. on Knowledge Discovery in Databases (KDD '05).

[3] Foussette, C., Hakenjos, D., Scholz, M.: KDD-Cup 2004: Protein Homology Task. ACM SIGKDD Explorations Newsletter **6** (2004) 128 – 131

[4] Schapire, R.E.: The Strength of Weak Learnability. Machine Learning **5** (1990) 197–227

[5] Freund, Y., Schapire, R.R.: A decision–theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences **55** (1997) 119 – 139

[6] Mitchell, T.M.: Machine Learning. McGraw Hill, New York (1997)

[7] Schapire, R.E., Freund, Y., Bartlett, P., Lee, S.: Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. Annals of Statistics (1998) 1651–1686

[8] Kivinen, J., Warmuth, M.K.: Boosting as Entropy Projection. In: Proc. of the 12th Conf. on Computational learning theory (COLT '99). 134 – 144

[9] Friedman, J.H., Hastie, T., Tibshirani, R.: Additive logistic regression: A statistical view of boosting. Annals of Statistics (2000) 337–374

[10] Klösgen, W.: Explora: A Multipattern and Multistrategy Discovery Assistant. In: Advances in Knowledge Discovery and Data Mining. AAAI Press/The MIT Press, (1996) 249–272

[11] Wrobel, S.: An Algorithm for Multi–relational Discovery of Subgroups. In: Principles of Data Mining and Knowledge Discovery (PKDD 97), 78–87

[12] Mackay, D.: Introduction To Monte Carlo Methods. In: Learning in Graphical Models. (1998) 175–204

[13] Schapire, R.E., Rochery, M., Rahim, M., Gupta, N.: Incorporating prior knowledge into boosting. In: Proc. of the 19th International Conference on Machine Learning. (2002)

[14] Mierswa, I., Klinkenberg, R., Fischer, S., Ritthoff, O.: A Flexible Platform for Knowledge Discovery Experiments: YALE – Yet Another Learning Environment. In: LLWA 03 - Tagungsband der GI-Workshop-Woche. (2003)

[15] Witten, I., Frank, E.: Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (2000)

[16] Blake, C., Merz, C.: UCI repository of machine learning databases (1998) http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[17] Lavrac, N., Flach, P., Zupan, B.: Rule Evaluation Measures: A Unifying View. In: 9th Int. Workshop on Inductive Logic Programming. LNCS, Springer (1999)