On Adaboost and Optimal Betting Strategies

Pasquale Malacaria[1] **and Fabrizio Smeraldi**[1]
[1]School of Electronic Engineering and Computer Science, Queen Mary University of London, London, UK

Abstract— *We explore the relation between the Adaboost weight update procedure and Kelly's theory of betting. Specifically, we show that Adaboost can be seen as an intuitive optimal betting strategy in the sense of Kelly. Technically this is achieved as the solution of the dual of the classical formulation of the Adaboost minimisation problem. Using this equivalence with betting strategies we derive a substantial simplification of Adaboost and of a related family of boosting algorithms. Besides allowing a natural derivation of the simplification, our approach establishes a connection between Adaboost and Information Theory that we believe may cast a new light on some boosting concepts.*

Keywords: Ensembles methods, Statistical Methods, Regression/Classification, Adaboost, Information theory, Optimal betting strategies

1. Introduction

Suppose you want to bet on a series of races of six horses, on which no information is initially available (the same horses participate in all races). The only information available after each race is in the form of the names of a subset of the losers for that race. This implies that you are not informed about the changes to your capital after each race, so your next bet can only be expressed as a percentage of your (unknown) capital. Let us furthermore assume that you are required to bet all of your capital on each race.

Initially, as nothing is known, it seems reasonable to spread one's bets evenly, so that your initial bet will allocate $1/6$-th of your capital on each horse.

Suppose now that you learn that horses 1,3, 4, 5 and 6 lost the first race. As your previous bet was optimal[1] (w.r.t. your information at the time) you want the new bet to be the "closest" possible to the previous bet, taking into account the outcome of the race. You could for instance decide to bet a fraction ρ of your capital on the latest "possible" winners (horse 1) and $1-\rho$ on the latest losers (horses 1,3,4,5,6). This ρ is a measure of how much you believe the result of the last race to be a predictor of the result of the next race.

Notice that, as you have to invest all your capital at each time, $\rho = 1$ (i.e. betting everything on the set of horses containing the previous winner) would be a bad choice: you would then lose all your capital if the following race is won by any other horse. Table 1 presents an algorithm implementing these ideas: bet $t+1$ on horse i is obtained from the previous bet t by multiplying it by $\frac{\rho}{\epsilon_t}$ (if the horse was a possible winner) and by $\frac{1-\rho}{(1-\epsilon_t)}$ (if it was a loser); ϵ_t is the total amount bet on the latest possible winners in the previous race t.

If you need to choose a value of ρ before all races, as no information is available, you may well opt for $\rho = 0.5$. This is a "non-committal" choice: you believe that the subset of losers of the each race revealed to you each time will not be a very good predictor for the result of the next race. In our example, where 1,3,4,5 and 6 lost the first race, choosing $\rho = 0.5$ will lead to the bets $(0.1, 0.5, 0.1, 0.1, 0.1, 0.1)$ on the second race. Table 2 shows the bets generated by applying the betting algorithm with $\rho = 0.5$ to a series of 4 races.

In Section 4 we show that the Adaboost weight update cycle is equivalent to this betting strategy with $\rho = 0.5$, modulo the trivial identification of the horses with training examples and of the possible winners with misclassified training data: this equivalence holds under an assumption of *minimal luck*, assumption which we will make more precise in section 5. As will be shown, this derives straightforwardly from a direct solution of the *dual* of the Adaboost minimisation problem. This result, that is the main contribution of our work, establishes the relation between Kelly's theory of optimal betting and Adaboost sketched in Figure 1. It also leads directly to the simplified formulation of Adaboost shown in Table 4.

The rest of this paper is organised as follows: Section 2 gives a brief introduction to the Adaboost algorithm; in Section 3 we outline the theory of optimal betting underlying the strategy displayed in Table 1. The equivalence between this betting strategy and Adaboost is proved in Section 4. In Section 5 we proceed in the opposite direction, and show how convergence theorems for Adaboost can be used to investigate the asymptotic behaviour of our betting strategy. Finally in Section 6 we generalize the correspondence between betting strategies and boosting and show that in the case of arbitrary choice of ρ we recover a known variant of Adaboost [1].

2. Background on Adaboost

Boosting concerns itself with the problem of combining several prediction rules, with poor individual performance,

[1]Optimal betting is here understood in the information theoretical sense. In probability terms betting all capital on the horse most likely to win will give the highest expected return, but will make you bankrupt with probability 1 in the long term. Maximizing the return has to be balanced with the risk of bankruptcy; see Section 3.

Table 1: The betting algorithm

Betting Algorithm, ρ is the fraction of the capital allocated to the latest possible winners:
Choose the initial bets $b_{1,i} = \frac{1}{m}$, where m is the number of horses;
For race number t:

1) Let W be the set of possible winners in the previous race. Compute $\epsilon_t = \sum_{i \in W} b_{t,i}$;
2) Update the bets for race $t+1$: $b_{t+1,i} = \frac{\rho}{\epsilon_t} b_{t,i}$ if $i \in W$, $b_{t+1,i} = \frac{1-\rho}{(1-\epsilon_t)} b_{t,i}$ otherwise.

into a highly accurate predictor. This is commonly referred to as combining a set of "weak" learners into a "strong" classifier. Adaboost, introduced by Yoav Freund and Robert Schapire [2], differs from previous algorithms in that it does not require previous knowledge of the performance of the weak hypotheses, but it rather adapts accordingly. This adaptation is achieved by maintaining a distribution of weights over the elements of the training set.

Adaboost is structured as an iterative algorithm, that works by maintaining a distribution of weights over the training examples. At each iteration a new weak learner (classifier) is trained, and the distribution of weights is updated according to the examples it misclassifies. The idea is to increase the importance of the training examples that are "difficult" to classify. The weights also determine the contribution of the each weak learner to the final strong classifier.

In the following, we will indicate the m training examples as $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$, where the $\mathbf{x}_i$ are vectors in a feature space X and the $y_i \in \{-1, +1\}$ are the respective class labels. Given a family of weak learners $\mathcal{H} = \{h_t : X \to \{-1, +1\}\}$, it is convenient to define $\mathbf{u}_{t,i} = y_i h_t(\mathbf{x}_i)$, so that $\mathbf{u}_{t,i}$ is $+1$ if h_t classifies $\mathbf{x}_i$ correctly, and -1 otherwise. If we now let $d_{t,i}$ be the distribution of weights at time t, with $\sum_i d_{t,i} = 1$, the cycle of iterations can be described as in Table 3.

Each iteration of the Adaboost algorithm solves the following minimisation problem [3], [4]:

$$\alpha_t = \arg\min_{\alpha} \log Z_t(\alpha), \tag{1}$$

where

$$Z_t(\alpha) = \sum_{i=1}^{m} d_{t,i} \exp(-\alpha u_{t,i}) \tag{2}$$

is the partition function that appears in point 3 of Table 3. In other words, the choice of the updated weights $d_{t+1,i}$ is such that α_t will minimise Z_t.

3. The gambler's problem: optimal betting strategies

Before investigating the formal relation between Adaboost and betting strategies, we review in this Section the basics of the theory of optimal betting, that motivated the heuristic reasoning of Section 1. Our presentation is a summary of the ideas in Chapter 6.1 of [5], based on Kelly's theory of gambling [6].

Let us consider the problem of determining the optimal betting strategy for a race run by m horses $x_1, \ldots, x_m$. For the i-th horse, let p_i, o_i represents respectively the probability of x_i winning and the *odds*, intended as o_i-for-one (e.g., at 3-for-1 the gambler will get 3 times his bet in case of victory).

Assuming that the gambler always bets all his wealth, let b_i be the fraction that is bet on horse x_i. Then if horse X wins the race, the gambler's capital grows by the *wealth relative* $S(X) = b(X)o(X)$. Assuming all races are independent, the gambler's wealth after n races is therefore $S_n = \Pi_{j=1}^{n} S(X_j)$.

It is at this point convenient to introduce the *doubling rate*

$$W(\mathbf{b}, \mathbf{p}) = E(\log S(X)) = \sum_{i=1}^{m} p_i \log(b_i o_i). \tag{3}$$

(in this information theoretical context, the logarithm should be understood to be in base 2). From the weak law of large numbers and the definition of S_n, we have that

$$\frac{1}{n} \log\ S_n = \frac{1}{n} \sum_{j=1}^{n} \log\ S(X_j) \to E(\log\ S(X)) \tag{4}$$

in probability, which allows us to express the gambler's wealth in terms of the doubling rate:

$$S_n = 2^{n \frac{1}{n} \log\ S_n} \doteq 2^{nE(\log\ S(X))} = 2^{nW(\mathbf{p}, \mathbf{b})} \tag{5}$$

where by "$\doteq$" we indicate equality to the first order in the exponent.

As can be shown the optimal betting scheme, i.e. the choice of the b_i that maximises $W(\mathbf{p}, \mathbf{b})$, is given by

Table 2: Bets b_i generated using the betting algorithm (Table 1) with $\rho = 1/2$ by information O_i over four races (w denotes a possible winner).

	x_1	x_2	x_3	x_4	x_5	x_6
b_0	0.1666	0.1666	0.1666	0.1666	0.1666	0.1666
O_0	ℓ	w	ℓ	ℓ	ℓ	ℓ
b_1	0.1	0.5	0.1	0.1	0.1	0.1
O_1	w	ℓ	ℓ	ℓ	w	ℓ
b_2	0.25	0.3125	0.0625	0.0625	0.25	0.0625
O_2	ℓ	ℓ	w	w	ℓ	ℓ
b_3	0.1428	0.1785	0.25	0.25	0.1428	0.0357
O_3	ℓ	w	ℓ	ℓ	ℓ	ℓ
b_4	0.0869	0.5	0.1521	0.1521	0.0869	0.0217

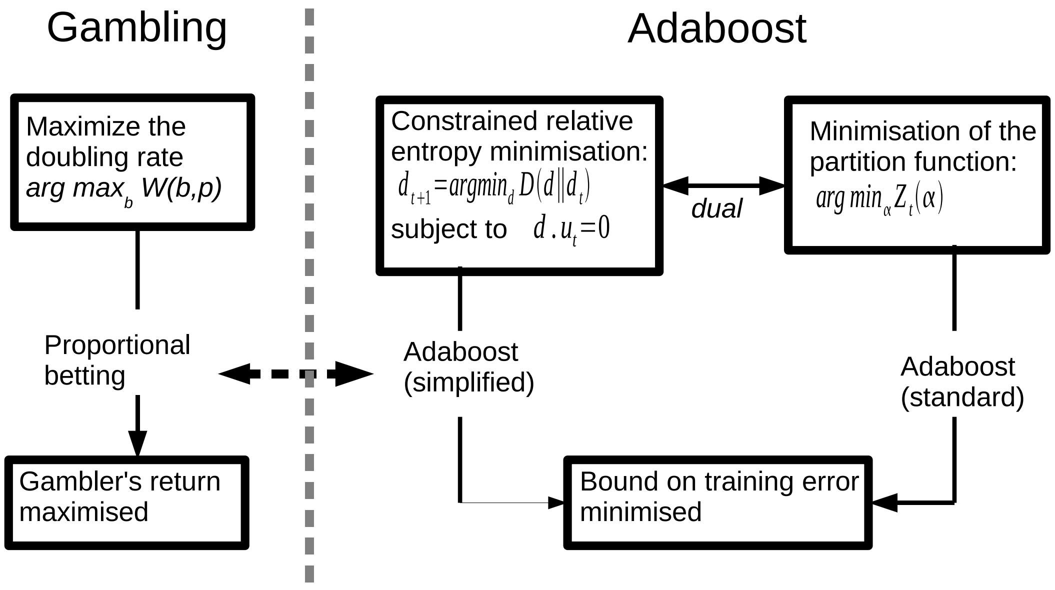
Gambling
Adaboost
Maximize the doubling rate arg max_b W(b,p)
Constrained relative entropy minimisation: d_{t+1}=argmin_d D(d||d_t) subject to d.u_t=0
dual
Minimisation of the partition function: arg min_α Z_t(α)
Proportional betting
Adaboost (simplified)
Adaboost (standard)
Gambler's return maximised
Bound on training error minimised

Figure 1: Relation between Kelly's theory of gambling and the standard and dual formulations of Adaboost.

Table 3: The standard formulation of the Adaboost algorithm

The Adaboost algorithm:

```
Initialise $d_{1,i} = \frac{1}{m}$.
For $t = 1, \ldots, T$:
    1) Select the weak learner $h_t$ that minimises the training error $\epsilon_t$, weighted by current distribution of weights $d_{t,i}$:
       $\epsilon_t = \sum_{i|u_{t,i}=-1} d_{t,i}$. Check that $\epsilon_t < 0.5$.
    2) Set $\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$
    3) Update the weights according to $d_{t+1,i} = \frac{1}{Z_t(\alpha_t)} d_{t,i}\exp(-\alpha_t u_{t,i})$, where $Z_t(\alpha_t) = \sum_{i=1}^m d_{t,i}\exp(-\alpha_t u_{t,i})$ is a
       normalisation factor which ensures that the $d_{t+1,i}$ will be a distribution.
Output the final classifier:
$H(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(x)\right)$
```

$\mathbf{b} = \mathbf{p}$, that is to say *proportional betting*. Note that this is independent of the odds o_i.

Incidentally, it is crucial to this concept of optimality that (4) is true with probability one in the limit of large n. Consider for example the case of 3 horses that win with probability $1/2$, $1/4$ and $1/4$ respectively and that are given at the same odds (say 3). In this case, it would be tempting to bet all our capital on the 1st horse, as this would lead to the highest possible expected gain of $3 \times (1/2) = 3/2$. However, it is easy to see that such a strategy would eventually make us bankrupt with probability one, as the probability that horse 1 wins the first n races is vanishingly small for large n, so that the expected value is not a useful criterion to guide our betting. Equation 4, instead, assures us that the capital will asymptotically grow like 2^{nW}. For the proportional betting strategy, in this case, $W = (1/2)\log(3\times 1/2) + (1/4)\log(3\times 1/4) + (1/4)\log(3\times 1/6) \approx 0.08$, so that the capital will grow to infinity with probability one [5].

In the special case of fair odds with $o_i = \frac{1}{p_i}$ (fair odds are defined as odds such that $\sum 1/o_i = 1$), the doubling rate becomes

$$W(\mathbf{p}, \mathbf{b}) = \sum p_i \log\left(\frac{b_i}{p_i}\right) = -D(\mathbf{p}||\mathbf{b})/\log_e 2 \qquad (6)$$

where $D(\mathbf{p}||\mathbf{b})$ is the Kullback-Leibler divergence (also known as the relative entropy) of the probability of victory $\mathbf{p}$ and the betting strategy $\mathbf{b}$.

Note that, since the odds are fair, the gambler can at best conserve his wealth, that is $\max_{\mathbf{b}} W(\mathbf{p}, \mathbf{b}) = 0$. In this case, it follows directly from the properties of the Kullback-Leibler divergence that the maximum occurs for $b_i = p_i$, i.e. that the optimal strategy is proportional betting.

In the following, for convenience, we will refer to Equation 6 as if the base of the logarithm were e instead than 2. This modification is inessential, and for the sake of convenience we will continue to refer to the new quantity as the "doubling rate".

4. Equivalence of Adaboost and betting strategy

We are now in a position to prove the equivalence of the betting algorithm introduced in Section 1 with the Adaboost weight update strategy, points 2) and 3) in Table 3. Modulo the identification of the horses with training examples and of the possible winners of each race with misclassified examples, this leads to the simplification of Adaboost shown in Table 4. Figure 1 illustrates schematically the relationships between these algorithms.

Our proof is based on the duality relations discussed in a number of excellent papers that cast Adaboost in terms of the minimisation of Kullback-Leibler or Bregman divergences [7], [8], [3]. In these, the Adaboost weight update equations are shown to solve the following problem:

$$\mathbf{d}_{t+1} = \arg\min_{\mathbf{d}} D(\mathbf{d}||\mathbf{d}_t) \quad \text{subject to}\, \mathbf{d}\cdot\mathbf{u}_t = 0. \tag{7}$$

In information theoretical terms, this can be interpreted as the Minimum Discrimination Information principle applied to the estimation of the distribution $\mathbf{d}_{t+1}$ given the initial estimate $\mathbf{d}_t$ and the linear constraint determined by the statistics $\mathbf{u}_t$ [9].

The key observation is now that the Kullback-Leibler divergence of the d_i and the $d_{t,i}$,

$$D(\mathbf{d}||\mathbf{d}_t) = \sum d_i \log\frac{d_i}{d_{t,i}}, \tag{8}$$

bears a strong formal analogy to Equation 6 above.

For the sake of readability, we discard the index t and simply write d_i for $d_{t,i}$ and $d_i^\star$ for $d_{t+1,i}$. The Adaboost dual problem Equation 7 then becomes

$$\mathbf{d}^\star = \arg\min_{\tilde{\mathbf{d}}} D(\tilde{\mathbf{d}}||\mathbf{d}) \qquad \text{subject to } \tilde{\mathbf{d}}\cdot\mathbf{u} = 0. \tag{9}$$

The constraint $\mathbf{d}^\star\cdot\mathbf{u} = 0$ can be rewritten as

$$\sum_{i|u_i=-1} d_i^\star = \sum_{i|u_i=+1} d_i^\star = 1/2. \tag{10}$$

With these notations, we are ready to prove the following

Theorem 1: The simplified weight update strategy in table 4 leads to the same weights as the original algorithm in Table 3.

Proof: we need to show that the weight update equations in table 4, that are the formal transcription of the betting strategy in Table 2, actually solve the optimisation problem Equation 9.

The constraint Equation 10 allows us to split the objective function in Equation 9 into the sum of two terms:

$$\begin{aligned} &D(\mathbf{d}^\star||\mathbf{d}) = \\ &= \sum_{i|u_i=-1} d_i^\star \log\frac{d_i^\star}{d_i} + \sum_{i|u_i=+1} d_i^\star \log\frac{d_i^\star}{d_i} \\ &= D_-(\mathbf{d}^\star,\mathbf{d}) + D_+(\mathbf{d}^\star,\mathbf{d}), \end{aligned} \tag{11}$$

each of which can be maximised separately.

Notice that neither of these terms taken separately represents a Kullback-Leibler divergence, as the partial sums over the $d_i^\star$ and d_i are not over distributions. However, this is easily remedied by normalising these sub-sequences separately. To this purpose, we introduce the Adaboost weighted error ϵ_t (see Table 3, point 1), that omitting the t index is equal to $\epsilon = \sum_{i|u_i=-1} d_i$. The first term of the summation can then be rewritten as

$$\begin{aligned} &D_-(\mathbf{d}^\star,\mathbf{d}) = \\ &= \frac{1}{2}\sum_{i|u_i=-1} 2d_i^\star \log\frac{2d_i^\star}{d_i/\epsilon} - \frac{1}{2}\log 2\epsilon \\ &= \frac{1}{2}D_-\left(2\mathbf{d}^\star||\mathbf{d}/\epsilon\right) - \frac{1}{2}\log 2\epsilon, \end{aligned} \tag{12}$$

where by D_- we mean the Kullback-Leibler divergence computed over the subset of indexes $\{i|u_i = -1\}$. The minimum point is now immediately evident: $2d_i^\star = d_i/\epsilon$ or, in the notation of Table 4,

$$d_{t+1,i} = \frac{1}{2\epsilon_t} d_{t,i} \qquad \text{if } u_{t,i} = -1. \tag{13}$$

Similarly, the minimization of D_+ leads to $2d_i^\star = d_i/(1-\epsilon)$, or

$$d_{t+1,i} = \frac{1}{2(1-\epsilon_t)} d_{t,i} \qquad \text{if } u_{t,i} = +1. \tag{14}$$

These are precisely the update equations given in Table 4, which proves the theorem.

We have thus shown that the simplified algorithm in Table 4 is completely equivalent to the original Adaboost algorithm in Table 3. Indeed, the two algorithms generate step by step the same sequence of weights, as well as the same decision function.

A similar simplification of Adaboost, although not so widely used, is already known to researchers in the field (see for instance [10]). However, to the best of our knowledge it has always been derived in a purely computational way by algebraic manipulation of the equations in Table 3. We believe that our derivation as a direct solution of the dual problem Equation 7 is both more straightforward and more illuminating, in that it brings out the direct correspondence with betting strategies represented in Figure 1. Finally, we note that a formally analogous weight update strategy has been employed in online *variants* of the algorithm [11], [12].

Table 4: Applying the proportional betting strategy to the solution of the Adaboost dual problem leads to a simpler formulation of the algorithm, equivalent to the original

```
Adaboost (simplified):

Initialise $d_{1,i} = \frac{1}{m}$.
For $t = 1, \ldots, T$:
    1) Select the weak learner $h_t$ that minimises the training error $\epsilon_t$, weighted by current distribution of weights $d_{t,i}$:
       $\epsilon_t = \sum_{i|u_{t,i}=-1} d_{t,i}$. Check that $\epsilon_t < 0.5$.
    2) Update the weights: $d_{t+1,i} = \frac{1}{2\epsilon_t} d_{t,i}$ if $u_{t,i} = -1$, $d_{t+1,i} = \frac{1}{2(1-\epsilon_t)} d_{t,i}$ otherwise.
Output the final classifier:
$H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$, with $\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$
```

5. Adaboost as a betting strategy

The considerations above suggest an interpretation of the betting algorithm in Table 1 as an iterative density estimation procedure in which an initial estimate, represented by the current bets $\mathbf{b}$, is refined using the information on the previous race. The updated estimate is then used to place new bets (in the notations of Table 1, $b_{t+1,i} = p_i$), in accordance to the proportional betting strategy.

This interpretation is supported by convergence results for the Adaboost weight update cycle, in either its standard form (Table 3) or its simplified form (Table 4). The application to our betting scenario with partial information is straightforward, modulo the the simple formal identifications $b_i = d_{t,i}$ and $p_i = d_{t+1,i}$.

To prove this, we will first state the following convergence theorem, which has been proved in [7], [8], [3]:

Theorem 2: If the system of the linear constraints has non-trivial solutions, i.e. there is some $\mathbf{d}$ such that $\mathbf{d} \cdot \mathbf{u}_t = 0$ $\forall t$, then the Adaboost weight distribution converges to the distribution $\mathbf{d}^\star$ satisfying all the linear constraints for which $D\left(\mathbf{d}^\star \| \mathbf{1}/m\right)$ is minimal (here by $\mathbf{1}$ we indicate the vector of all ones).

In other words, the weight update algorithm converges to a probability density that is, among all those that satisfy all the constraints, the closest to the original guess of a uniform distribution. This "approximation" property is best expressed by the following "Pythagorean theorem" for Kullback-Leibler divergences, that was derived in [13] in the context of an application to iterative density estimation (a generalisation to Bregman divergences can be found in [8]). For all solutions $\mathbf{d}$ of the system of linear constraints, we have

$$D\left(\mathbf{d} \| \mathbf{1}/m\right) = D\left(\mathbf{d} \| \mathbf{d}^*\right) + D\left(\mathbf{d}^* \| \mathbf{1}/m\right). \quad (15)$$

In terms of our gambling problem, theorem 2 above means that the betting algorithm in Table 1 converges to an estimate $\mathbf{p}^\star = \mathbf{b}^\star$ of the probability of winning of each single horse. Among all the densities that satisfy the constraints imposed by the information on the previous races, $\mathbf{p}^\star$ is the one that would maximise the doubling rate of the initial uniform bet, $W\left(\mathbf{p}^\star, \mathbf{1}/m\right)$ (strictly speaking, theorem 2 proves this result only for the case that $\rho = 1/2$; however, generalisation to the case of a generic ρ is straightforward, see [7]).

However, for convergence to occur the information $\mathbf{u}_t$ on the various races must be provided in the order determined by Point 1) in either Table 3 or Table 4, corresponding to the Adaboost requirement that the weak learner should minimise the error with respect to the weights. In the betting case, of course, this appears problematic as the gambler has no influence on the order in which the results occur. The corresponding assumption corresponding to the minimal error requirement would be, in a sense, one of "minimal luck": that is, once we decide on a betting strategy, the next race is the one for which the set of the possible winners has, according to our estimate, the lowest total probability.

When the conditions for convergence are met, the "Pythagorean" equality Equation 15 quantifies the advantage of the estimated probability $\mathbf{p}^\star$ over any other density $\mathbf{p}$ compatible with the constraint, if the initial uniform distribution (or indeed any other density obtained in an intermediate iteration) are used for betting:

$$W\left(\mathbf{p}, \mathbf{1}/m\right) = W\left(\mathbf{p}^*, \mathbf{1}/m\right) - D\left(\mathbf{p} \| \mathbf{p}^*\right). \quad (16)$$

6. Extension of the simplified approach to Adaboost-ρ

In the following, we generalize the simplified approach outlined in Table 4. In terms of the betting strategy described in the introduction we are now considering a gambler using a value of ρ different from $1/2$. For simplicity we will consider $0 < \rho \leq 1/2$; it is an easy observation that the case $1 > \rho \geq 1/2$ is obtained by symmetry. This generalised constraint produces an algorithm known as Adaboost-ρ, that has been introduced in [1]. This variant of Adaboost has an intuitive geometric and statistical interpretation in terms of handling a degree of correlation between the new weights and the output of the weak learner. It also is an ideal testbed to show how our simplified approach can lead to straightforward implementations of generalised versions of Adaboost. The question of whether an extension of our approach to the entire class of algorithms described in [7] is feasible is left for future work.

Similarly to what has been done for Adaboost, we directly solve the generalised dual optimisation problem

$$\mathbf{d}_{t+1} = \arg\min_{\mathbf{d}} D(\mathbf{d}^\star||\mathbf{d}_t) \qquad \text{subject to } \mathbf{d}^\star\cdot\mathbf{u}_t = 1-2\rho. \tag{17}$$

where $0 < \rho \leq 1/2$. The solution to this problem will provide the simplified weight update rules (that this is indeed the dual of the Adaboost-ρ problem will be clarified in Section 6.1).

Proceeding as in Section 4, we simplify the notation by omitting the index t in $d_{t,i}$. Equation (17) implies the constraints

$$\sum_{i|u_i=-1} d_i^\star = \rho, \qquad \sum_{i|u_i=+1} d_i^\star = 1-\rho, \tag{18}$$

The objective function Equation 17 can again be decomposed as done in Equation 11:

$$D(\mathbf{d}^\star, \mathbf{d}|\rho) = D_-(\mathbf{d}^\star, \mathbf{d}|\rho) + D_+(\mathbf{d}^\star, \mathbf{d}|\rho), \tag{19}$$

where D_- and D_+ account for the misclassified and the correctly classified training examples respectively.

Writing out the right hand side explicitly yields

$$\begin{aligned} D_-(\mathbf{d}^\star, \mathbf{d}|\rho) &= \rho \sum_{i|u_i=-1} \frac{d_i^\star}{\rho} \log \frac{d_i^\star/\rho}{d_i/\epsilon} - \rho \log\frac{\epsilon}{\rho} = \\ &= \rho D_- \left(\frac{\mathbf{d}^\star}{\rho} \middle\| \frac{\mathbf{d}}{\epsilon} \right) - \rho\log\frac{\epsilon}{\rho} \end{aligned}$$

and similarly

$$D_+(\mathbf{d}^\star, \mathbf{d}|\rho) = (1-\rho) D_+ \left(\frac{\mathbf{d}^\star}{1-\rho} \middle\| \frac{\mathbf{d}}{1-\epsilon} \right) - (1-\rho)\log\frac{1-\epsilon}{1-\rho}$$

where again D_- and D_+ stand for the Kullback-Leibler divergences calculated over the subsets of indexes $\{i|u_i = \mp 1\}$. It follows straightforwardly that the minimum is achieved for $d_i^\star = \frac{\rho}{\epsilon} d_i$ if $u_i = -1$, and $d_i^\star = \frac{1-\rho}{1-\epsilon} d_i$ if $u_i = +1$. These effectively are the weight update rules for the generalised algorithm, as shown in Table 5.

6.1 Standard solution of the Adaboost-ρ problem

For comparison, we derive here the standard version of the Adaboost-ρ algorithm in the notations used in this paper (apart for trivial changes of notation, this is the algorithm originally published in [1]).

We first need to obtain the objective function for the minimisation problem associated to Adaboost-ρ, of which Equation 17 is the dual. In other words, we need to determine a suitable function $Z_{t|\rho}(\alpha)$ which generalises the function in Equation 2, that is to say,

$$\begin{aligned} -\min_{\mathbf{d}|\mathbf{d}\cdot\mathbf{u}_t=0} D(\mathbf{d}||\mathbf{d}_t) &= \rho\log\frac{\epsilon}{\rho} + (1-\rho)\log\frac{1-\epsilon}{1-\rho} && (20) \\ &= \min_{\alpha} \log Z_{t|\rho}(\alpha). && (21) \end{aligned}$$

Following the construction outlined in [7] and essentially setting $M_{ij} = u_{j,i} + 2\rho - 1$ in the Lagrange transform, we find

$$Z_{t|\rho}(\alpha) = \sum_{i=1}^{m} d_{t,i} \exp\left(-\alpha(u_{t,i} + 2\rho - 1)\right), \tag{22}$$

that is minimised by

$$\alpha_{t|\rho} = \frac{1}{2}\log\left(\frac{\rho}{1-\rho}\frac{1-\epsilon_t}{\epsilon_t}\right). \tag{23}$$

By substituting these values in the equation below it is easy to verify directly that

$$d_{t+1,i} = \frac{\exp\left(-\alpha_{t|\rho}(u_{t,i} + 2\rho - 1)\right)}{Z_{t|\rho}\left(\alpha_{t|\rho}\right)} d_{t,i} \tag{24}$$

yields the same weight update rules as given in Table 5, and is indeed the direct generalisation of the update rules in the standard formulation of Adaboost given in Table 3.

The complete standard Adaboost-ρ algorithm is shown in Table 6. Again, we note how the dual solution we proposed in Table 5 presents a considerable simplification of the standard formulation of the algorithm.

7. Conclusions

We discussed the relation between Adaboost and Kelly's theory of gambling. Specifically, we showed how the dual formulation of Adaboost formally corresponds to maximising the doubling rate for a gambler's capital, in a partial information situation. The applicable modification of the optimal proportional betting strategy therefore maps to a direct, intuitive solution of the dual of the Adaboost problem. This leads naturally to a substantial formal simplification of the Adaboost weight update cycle. Previous derivation of similar simplifications are, to the best of our knowledge, purely computational, and therefore, besides being more cumbersome, lack any meaningful interpretation.

The implications of the correspondence between Adaboost and gambling theory works in both directions; as we have shown, it is for instance possible to use convergence results obtained for Adaboost for investigating the asymptotic behaviour of the betting strategy we introduced in Table 1.

We believe that a further investigation of the relation between Adaboost and Kelly's theory, or indeed information theory in general, may lead to a deeper insight into some boosting concepts, and possibly a cross-fertilization between these two domains.

```
**Adaboost-$\rho$ simplified:**

Initialise $d_{1,i} = \frac{1}{m}$.
For $t = 1, \dots, T$:
  1) Select the weak learner $h_t$ that minimises the training error $\epsilon_t$, weighted by current distribution of weights $d_{t,i}$:
     $\epsilon_t = \sum_{i|u_{t,i}=-1} d_{t,i}$. Check that $\epsilon_t < \rho$.
  2) Update the weights: $d_{t+1,i} = \frac{\rho}{\epsilon_t} d_{t,i}$ if $u_{t,i} = -1$, $d_{t+1,i} = \frac{1-\rho}{1-\epsilon_t} d_{t,i}$ otherwise.
Output the final classifier:
$H_\rho(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_{t|\rho} h_t(\mathbf{x})\right)$, with $\alpha_{t|\rho} = \frac{1}{2}\log\left(\frac{\rho}{1-\rho}\frac{1-\epsilon_t}{\epsilon_t}\right)$
```

Table 5: Adaboost-ρ generalises the Adaboost weight update constraint to the case $\mathbf{d} \cdot \mathbf{u} = 1 - 2\rho$, with $0 < \rho \leq 1/2$

```
**The Adaboost-$\rho$ algorithm:**

Initialise $d_{1,i} = \frac{1}{m}$.
For $t = 1, \dots, T$:
  1) Select the weak learner $h_t$ that minimises the training error $\epsilon_t$, weighted by current distribution of weights $d_{t,i}$:
     $\epsilon_t = \sum_{i|u_{t,i}=-1} d_{t,i}$. Check that $\epsilon_t < \rho$.
  2) Set $\alpha_{t|\rho} = \frac{1}{2}\log\left(\frac{\rho}{1-\rho}\frac{1-\epsilon_t}{\epsilon_t}\right)$.
  3) Update the weights according to $d_{t+1,i} = \frac{\exp\left(-\alpha_{t|\rho}(u_{t,i}+2\rho-1)\right)}{Z_{t|\rho}\left(\alpha_{t|\rho}\right)} d_{t,i}$,
     where $Z_{t|\rho}(\alpha_{t|\rho}) = \sum_{i=1}^{m} d_{t,i} \exp\left(-\alpha_{t|\rho}(u_{t,i} + 2\rho - 1)\right)$, is a normalisation factor which ensures that the $d_{t+1,i}$ will
     be a distribution.
Output the final classifier:
$H_\rho(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_{t|\rho} h_t(x)\right)$
```

Table 6: The standard formulation of the Adaboost-ρ algorithm

References

[1] G. Rätsch, T. Onoda, and K.-R. Muller, “Soft margins for adaboost,” *Machine Learning*, vol. 42, no. 3, pp. 287–320, 2001. [Online]. Available: citeseer.ist.psu.edu/657521.html

[2] Y. Freund and R. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Science*, vol. 55, no. 1, 1997.

[3] M. W. J. Kivinen, “Boosting as entropy projection,” in *Proceedings of COLT’99*. ACM, 1999.

[4] R. Schapire and Y. Singer, “Improved boosting algorithms using confidence-rated predictions,” *Machine Learning*, vol. 37, no. 3, 1999.

[5] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley Interscience, 1991.

[6] J. J. L. Kelly, “A new interpretation of information rate,” *Bell System Technical Journal*, vol. 35, pp. 917–926, July 1956.

[7] M. Collins, R. E. Schapire, and Y. Singer, “Logistic regression, adaboost and bregman distances,” in *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, 2000.

[8] S. D. Pietra, V. D. Pietra, and H. Lafferty, “Inducing features of random field,” *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, April 1997.

[9] S. Kullback, *Information Theory and Statistics*. Wiley, 1959.

[10] C. Rudin, R. E. Schapire, and I. Daubechies, “On the dynamics of boosting,” in *Advances in Neural Information Processing Systems*, vol. 16, 2004.

[11] N. C. Oza and S. Russell, “Online bagging and boosting,” in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, 2005, pp. 2340–2345.

[12] H. Grabner and H. Bischof, “On-line boosting and vision,” in *Proceedings of CVPR*, vol. 1, 2006, pp. 260–267.

[13] S. Kullback, “Probability densities with given marginals,” *The Annals of Mathematical Statistics*, vol. 39, no. 4, 1968.