Lossless or Quantized Boosting with Integer Arithmetic

Richard Nock [1 2 3] **Robert C. Williamson** [2 1]

Abstract

In supervised learning, efficiency often starts with the choice of a good loss: support vector machines popularised Hinge loss, Adaboost popularised the exponential loss, etc. Recent trends in machine learning have highlighted the necessity for training routines to meet tight requirements on communication, bandwidth, energy, operations, encoding, among others. Fitting the often decades-old state of the art training routines into these new constraints does not go without pain and uncertainty or reduction in the original guarantees. Our paper starts with the design of a new strictly proper canonical, twice differentiable loss called the Q-loss. Importantly, its mirror update over (arbitrary) rational inputs uses only integer arithmetics – more precisely, the sole use of $+, -, /, \times, |.|$. We build a learning algorithm which is able, under mild assumptions, to achieve a lossless boosting-compliant training. We give conditions for a quantization of its main memory footprint, weights, to be done while keeping the whole algorithm boosting-compliant. Experiments display that the algorithm can achieve a fast convergence during the early boosting rounds compared to AdaBoost, even with a weight storage that can be 30+ times smaller. Lastly, we show that the Bayes risk of the Q-loss can be used as node splitting criterion for decision trees and guarantees optimal boosting convergence.

1. Introduction

More often than not, today's supervised machine learning (ML) toolkit exploits ideas and algorithms grounded in decades-old approaches: stochastic gradient descent (SGD, Robbins & Monro (1951)), backpropagation (Rumelhart et al., 1986), boosting (Freund & Schapire, 1997), support vector machines (Boser et al., 1992), etc. . To be carried out in today's settings, such long-lived algorithms require care and caution to cope with practical constraints on parallel and distributed processing, memory, communication, bandwidth or energy, the availability of reduced sets of operations, the coding size of models, etc. These constraints are non exclusive and arise for a number of reasons that were just mostly not in the original bills of specifications for those algorithms, essentially "reduced" to the optimisation of a risk (Vapnik, 1998). Many started to be formally studied a decade later (Bottou & Bousquet, 2007), only to become almost a further decade later a focus of attention for the whole field.

To adapt, tailor and run such algorithms, three important tools (not independent from each other) emerge that we refer to as (i) `quantization`, (ii) `encoding` and (iii) `operations`. (i-ii) deal with parameters stored: in (i), we limit the number of different values (and eventually the values) that may be coded; in (ii) we tune the encoding of those values. In (iii) we control the set of operations used to train or classify. Tools from (i-iii) can cover the complete training or classification phases (See section 2). Most of the state of the art has been focusing in using (i-iii) for adapting the aforementioned algorithms to today's constraints. While this has met with unquestionable empirical success, there has been comparatively little formal understanding of the impacts of changes; furthermore, the resulting algorithms are not a panacea for such constraints, to the extent that there are calls to redesign learning algorithms when possible to be constraint friendly from the start (Goldwasser, 2018).

Such is the starting point of our contribution: we design a new boosting algorithm for ensembles, compliant with the original boosting theory requirements (Kearns, 1988), with the following characteristics with respect to (i-iii) above:
(i) `quantization`: conditions are given for boosting weights to be quantized without loosing boosting-compliance;
(ii) `encoding`: if the inputs are rational then the full algorithm and the output model's classification can be carried out with no precision loss using integer arithmetic;
(iii) `operations`: the algorithm just needs $+, -, \times, /, |.|$.

In other words, under mild conditions, the algorithm is able to compute a *lossless* solution to the boosting problem. Popular boosting algorithms like AdaBoost (Freund & Schapire,

[1]Data61, [2]The Australian National University, Canberra & [3]The University of Sydney, Sydney (Australia). Correspondence to: <richard.nock@data61.csiro.au, bob.williamson@anu.edu.au>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

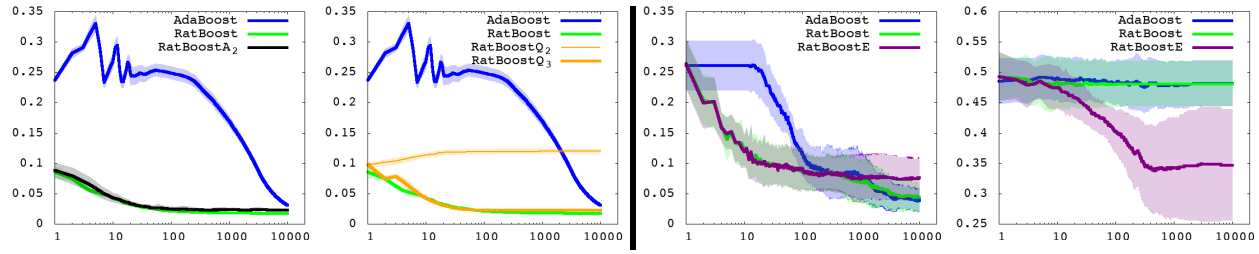

Figure 1: *Left pane*: test error (y, $\pm$ standard deviation) on UCI domain Hardware vs iteration number t (x, logscale), for AdaBoost (Schapire & Singer, 1999) (blue) vs RATBOOST (green). We observe that RATBOOST tends to converge faster to good solutions during early iterations. We plot in addition two results for quantized versions of RATBOOST: adaptive with 2 bits index (left, black) and regular with 2 and 3 bits indexes (right, orange). A 2 bits index represents a reduction in weight storage by a factor > 30 compared to RATBOOST or Adaboost. *Right pane*: additional results on two UCI domains (left: mice, right: yeast); RATBOOSTE (magenta) implements an integer arithmetic version of RATBOOST.

1997) are not compatible with constraints (i) and (ii), and we do not know any result on (iii), which is a non-trivial but major problem. Indeed, weight handling has been known to be a source of numerical precision errors for decades (Kohavi, 1998). Furthermore, weights represent the main memory overhead and become a significant communication overhead too for distributed boosting (Lazarevic & Obradovic, 2002). Finally, numbers encoded in a computer are finite and inevitably quantized (Widrow & Kollár, 2008), so our contribution on (i) extends beyond the scope of our paper.

To design such an algorithm, we start from the root of any supervised learning algorithm: we design a *loss function*, called the Q-loss, having (a) the desirable statistical property of being strictly proper canonical, (b) a surrogate with the desirable optimisation property of being strictly convex and twice differentiable, (c) a mirror update which entails only $+, -, \times, /, |.|$ and (d) a canonical link whose image spans the full $\mathbb{R}$. For example, (a) rules out the exp-loss, (c) rules out the log-loss and Matsushita loss, (d) rules out the square loss. While (c) has clear importance in our case, all others are crucial as well: for example, the fact that the square loss fails at (d) is a source of tricky bounding problems (Friedman et al., 2000). Since the mirror update for a loss can be of use in various other settings, such as on-line learning, our contribution on the Q-loss extends as well beyond the batch learning scope of our paper.

We obtain a boosting algorithm that we call RATBOOST. To get an idea of the performances of RATBOOST, Figure 1 (left pane) shows an example of run on an UCI domain (Blake et al., 1998) comparing three flavours of RATBOOST to AdaBoost (Schapire & Singer, 1999) to learn linear models. All algorithms are run for 10 000 iterations. RATBOOSTQ$_b$ is RATBOOST with a deterministic regular quantization for weights with a b-bits index (we encode $2^b - 1$ weights), and RATBOOSTA$_b$ is the same as RATBOOSTQ$_b$ but with an adaptive quantization scheme for weights (see §6). We see that RATBOOST beats AdaBoost significantly over the first boosting rounds, but also while 2-bit deterministic regular quantization of weights fails to converge (to good values) for RATBOOSTQ$_.$, 3-bit does, as well as 2-bit *adaptive* quantization. In this last case, we achieve a $\sim$31.9$\times$ reduction in coding size for weights over RATBOOST or AdaBoost.

While our focus is on ensembles, we also show an additional result on the boosting ability of domain partitioning classifiers using the Bayes risk of the Q-loss. This is the state of the art approach to learning decision trees Quinlan (1993); Schapire & Singer (1999); Dietterich et al. (1996); Kearns & Mansour (1996). We show that the Q-loss displays *optimal* convergence rates in this case, beating the guarantees of CART and C4.5 (Breiman et al., 1984; Quinlan, 1993).

The rest of our paper is as follows: §2 summarizes related results, §3 presents definitions, §4 presents the Q-loss and its properties, §5 presents boosting results for ensembles, §6 summarizes experiments, §7 presents boosting results for decision trees and a last section discusses and concludes. All proofs, detailed experiments are given in an SM.

2. Related work

A substantial body of work has essentially focused on `quantization` for architectures (often deep learning) (Banner et al., 2018; Gupta et al., 2015; Hubara et al., 2016) and algorithms: stochastic gradient descent (Bernstein et al., 2018; Alistarh et al., 2017) (and references therein), support vector machines (Qin et al., 2014; Sakr et al., 2017b) and regression (Slawski & Li, 2015); in that last case, the theory builds on contributions from sparse signal recovery, whose motivations are loosely related to ours (see references in Slawski & Li (2015)). Three categories of `quantization` have been investigated so far: deterministic and fixed (*e.g.* regular) (Gupta et al., 2015), fixed but *stochastic* (Gupta et al., 2015), and finally *adaptive*, essentially used in two previous work: Jacob et al. (2018) fits the quantization range dynamically and Zhang et al. (2017) learn the quantization with bounds on quantization error. In the stochastic case, the range of values is fixed but a sochas-

tic assignation of the quantized value is carried out to ensure that the expected quantized value is the true value. Early approaches on distributed boosting can be related to an early form of `quantization` (Lazarevic & Obradovic, 2002).

The topic of `encoding` is receiving substantial attention because of its huge interest and connections to hardware. Progress has been essentially been experimental and met with mixed outcomes (Drumond et al., 2018; Narang et al., 2018) with reductions from floating to fixed point (Sakr et al., 2017a; Lin et al., 2016) or hybrid/mixed floating point (Drumond et al., 2018; Narang et al., 2018) and with substantial effort, mixed float/integer arithmetics (Jacob et al., 2018). The experimental motives behind `encoding` are clear: a careful encoding allows the reduction of time, space, bandwidth, energy in proportions that can *exceed* the reduction factor in encoding (Drumond et al., 2018; Narang et al., 2018). Finally, `operations` is strongly motivated by hardware considerations (Jacob et al., 2018). It is also fundamental for private learning (Goldwasser, 2018), in particular with the development of training algorithms on top of *partially* homomorphic encryption systems allowing a subset of arithmetic operations (Hardy et al., 2017).

Even when related results were obtained decades ago (Dasgupta et al. (1990) and references therein), formal results are still comparatively sparse in the field and include the fixed-point approaximation of a model (Sakr et al., 2017a), signal recovery from b-bits linear regression (Slawski & Li, 2015), precision requirements for linear SVM (Sakr et al., 2017b), and 1-bit SGD compression (Bernstein et al., 2018).

3. Definitions

▷ **Batch learning**: we use the shorthand notations $[n] \doteq \{1, 2, ..., n\}$ for $n \in \mathbb{N}_*$ and $z \cdot [a, b] \doteq [za, zb]$ for $z \geq 0, a \leq b \in \mathbb{R}$. We also let $\overline{\mathbb{R}} \doteq [-\infty, \infty]$. In the batch supervised learning setting, one is given a training set of m examples $S \doteq \{(\boldsymbol{x}_i, y_i), i \in [m]\}$, where $\boldsymbol{x}_i \in \mathcal{X}$ is an observation ($\mathcal{X}$ is called the domain: often, $\mathcal{X} \subseteq \mathbb{R}^d$) and $y_i \in \mathcal{Y} \doteq \{-1, 1\}$ is a label, or class. The objective is to learn a *classifier*, *i.e.* a function $h : \mathcal{X} \to \mathbb{R}$ which belongs to a given set $\mathcal{H}$. The goodness of fit of some h on $\mathcal{S}$ is evaluated by a given *loss* $F(h; \mathcal{S})$:

$$F(h; \mathcal{S}) \quad \doteq \quad \mathbb{E}_S[F(yh(\boldsymbol{x}))], \tag{1}$$

where $F : \mathbb{R} \to \mathbb{R}$. F shall be used without ambiguity to denote both the (expected) loss in the left hand side and the function F. The most popular example of such a function F is probably the 0/1-loss, $F_{0/1}(z) \doteq 1_{z \leq 0}$, where $1_{\cdot}$ is the indicator function. In this case, (1) is the *empirical risk*.

▷ **Proper losses**: general criteria such as (1) are also called *surrogate criteria* (Buja et al., 2005), either because they define upperbounds of the 0/1-loss or because they are optimisation devices solving in disguise the *class probability estimation* problem for $\mathbb{P}[\mathsf{Y} = 1|\mathsf{X}]$ via classifier h as estimate $\hat{\mathbb{P}}[\mathsf{Y} = 1|\mathsf{X}, h]$. In that latter case, losses whose minimum is achieved by Bayes rule are called *proper* (and *strictly* if the minimizer is unique) and the subset of which whose estimation $\hat{\mathbb{P}}[\mathsf{Y} = 1|\mathsf{X}, h]$ is computed in disguise directly from the loss itself are called *canonical*. A formal definition of proper canonical losses shall be given below; popular losses like the logistic and square loss are proper canonical, but Adaboost's exponential loss is not proper canonical (Buja et al., 2005, Section 16). We now review the main properties of these losses, adopting notations of (Reid & Williamson, 2010). A loss for class probability estimation is a function $\ell : \mathcal{Y} \times (0, 1) \to \mathbb{R}$ whose expression can be split according to *partial* losses ℓ_1, ℓ_{-1},

$$\ell(y, u) \quad \doteq \quad [\![y = 1]\!] \cdot \ell_1(u) + [\![y = -1]\!] \cdot \ell_{-1}(u), \tag{2}$$

where $[\![P]\!]$ is the truth value of a predicate P. We extend the loss to $\mathcal{Y} \times [0, 1]$ by taking the limits. The loss is *symmetric* when $\ell_1(u) = \ell_{-1}(1 - u), \forall u \in (0, 1)$ (Nock & Nielsen, 2008). Its *pointwise* risk is

$$L(\pi, u) \quad \doteq \quad \mathbb{E}_{\mathsf{Y} \sim \mathrm{B}(\pi)}[\ell(\mathsf{Y}, u)], \tag{3}$$

where $\mathrm{B}(\pi)$ refers to a Bernoulli for picking $\mathsf{Y} = 1$. The associated pointwise *Bayes* risk is defined as

$$\underline{L}(\pi) \quad \doteq \quad \inf_u L(\pi, u). \tag{4}$$

The loss is called *proper* iff π realizes the $\inf$ in (4) and *strictly proper* if it is the only argument realizing the $\inf$.

Example 1 *The square loss has* $\ell_1^{\mathrm{SQ}}(u) \doteq (1/2) \cdot (1 - u)^2$ *and* $\ell_{-1}^{\mathrm{SQ}}(u) \doteq (1/2) \cdot u^2$. *Its Bayes risk is defined by Gini entropy,* $\underline{L}^{\mathrm{SQ}}(\pi) = (1/2) \cdot \pi(1 - \pi)$. *The logistic loss has* $\ell_1^{\mathrm{LOG}}(u) \doteq -\log u$ *and* $\ell_{-1}^{\mathrm{LOG}}(u) \doteq -\log(1 - u)$. *Its Bayes risk is the binary entropy,* $\underline{L}^{\mathrm{LOG}}(u) = -\pi \log(\pi) - (1 - \pi) \log(1 - \pi)$. *Both losses are strictly proper.*

▷ **Link functions**: the connection between class probability estimation and real valued prediction through classifier h in the loss in (1) is achieved through a *link function*, that is, an invertible function $\psi : [0, 1] \to \overline{\mathbb{R}}$ (Buja et al., 2005; Reid & Williamson, 2010). One such link is fundamental for a loss function, its *canonical link*. (Buja et al., 2005) define the canonical link ψ_ℓ (ψ for short when there is no ambiguity on the loss) of a loss ℓ as

$$\psi_\ell(u) \quad \doteq \quad -\underline{L}' + K, \tag{5}$$

where $'$ denotes derivative and K is a constant. We assume that $K = 0$. This choice can be formally grounded (Nock & Nielsen, 2008, Section 5), but it is also intuitive: when the loss is symmetric, $\psi(1/2) = 0$, which is a meaningful way to associate to the most uncertain class probability ($\pi = 1/2$) the most unconfident real valued prediction (0).

Example 2 *The canonical link for the logistic loss is given by* $\psi_{\text{LOG}}(u) = \log(u/(1-u))$*; the canonical link for the square loss is given by* $\psi_{\text{SQ}}(u) = u - 1/2$*.*

▷ **Convex surrogates**: there is an interesting connection between a subset of losses as in (1) and proper losses via canonical links (Buja et al., 2005; Nock & Nielsen, 2008; Reid & Williamson, 2010). Assuming from now on that $K = 0$ in (5), we can craft, from any proper symmetric loss, a *balanced convex loss* (Nock & Nielsen, 2008) $F_\ell : \mathbb{R} \to \mathbb{R}$ (F for short when there is no ambiguity), as:

$$F_\ell(z) \doteq (-\underline{L})^\star(-z), \quad (6)$$

where $\star$ denotes the Legendre conjugate of F, $F^\star(z) \doteq \sup_{z' \in \text{dom}(F)} \{zz' - F(z')\}$ (Boyd & Vandenberghe, 2004). For simplicity, we just call F_ℓ the convex surrogate of ℓ. F_ℓ is important because its minimization amounts to the minimization of the pointwise risk in (3) via the canonical link (Nock & Nielsen, 2008, Lemma 1), as follows. The derivative of F_ℓ involves the canonical link (Boyd & Vandenberghe, 2004; Reid & Williamson, 2010):

$$F'_\ell(z) = -(-\underline{L}')^{-1}(-z) = 1 - \psi^{-1}(z). \quad (7)$$

The class probability estimator (also called the *matching prediction* for z given ψ)

$$\hat{\mathbb{P}}[y = 1|\psi; z] = \psi^{-1}(z) \quad (8)$$

gives the expression that ties the minimization of F_ℓ (for $z \in \mathbb{R}$) to the minimization of the sample-wise expected risk in (3) (for $u \in [0, 1]$) (Nock & Nielsen, 2008). In short, the minimization of $F_\ell(h; \mathcal{S})$ in (1) achieves in disguise the minimization of the *full* expected pointwise risk:

$$\mathbb{L}(u; \pi, M) \doteq \mathbb{E}_{\mathsf{X} \sim M}[L(\pi(\mathsf{X}), u(\mathsf{X}))], \quad (9)$$

where M is the empirical marginal of X, $\pi(\mathsf{X}) \doteq \mathbb{P}_{\mathcal{S}}[y = 1|\mathsf{X}]$ and $u(\mathsf{X}) \doteq \hat{\mathbb{P}}[y = 1|\psi; h(\mathsf{X})]$ as defined in (8) where ψ is the canonical link in (5). Any loss in (2) for which this decomposition holds for the full expected pointwise risk is called *proper canonical*.

Example 3 *The matching prediction for the logistic loss is given by* $\hat{\mathbb{P}}[y = 1|\psi_{\text{LOG}}; z] = 1/(1 + \exp(-z))$*. In theory, the matching prediction for the square loss,* $\hat{\mathbb{P}}[y = 1|\psi_{\text{SQ}}; z] = z + \frac{1}{2}$*, constrains the domain of* $z \in \frac{1}{2} \cdot [-1, 1]$*.*

4. The Q-loss

We first define the Q-loss.

Definition 4 *The Q-loss* ℓ^Q *is defined for any* $\varepsilon \in (0, 1/2)$*,* $\varrho > 0$ *(implicit) by the following partial losses:*

$$\ell^Q_y(u) \doteq \varrho \cdot \left(\log\left(\frac{u}{\varepsilon}\right) + [\![y = 1]\!] \cdot \left(-2 + \frac{1}{u}\right)\right) \text{ if } u \leq 1/2, \quad (10)$$

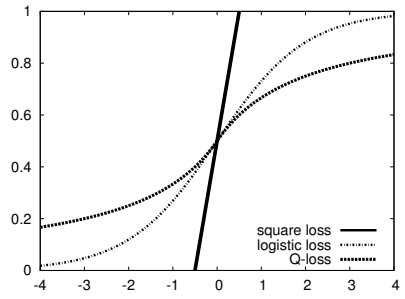

Figure 2: Matching prediction $\hat{\mathbb{P}}[y = 1|\psi; z]$ obtained using (8) for ψ being the canonical link of three proper losses. Remark that the domain of z is $\mathbb{R}$ for both the logistic and our Q-loss, but restricted to $\frac{1}{2} \cdot [-1, 1]$ for the square loss.

and $\ell^Q_y(u) \doteq \ell^Q_{-y}(1-u)$ *if* $u > 1/2$*.*

We define two additional functions, $\text{H}(z) \doteq 0 \vee -z$ and $\text{err}(u) \doteq u \wedge (1-u)$, where $\vee, \wedge$ are shorthands for the max and min, respectively. We are now ready to show the key properties of the Q-loss.

Theorem 5 *The Q-loss is strictly proper and symmetric. Its pointwise Bayes risk is:*

$$\underline{L}^Q(u) = \varrho \cdot \left(\log\left(\frac{\text{err}(u)}{\varepsilon}\right) + 1 - 2 \cdot \text{err}(u)\right). \quad (11)$$

Its canonical link and inverse canonical link are:

$$\psi^Q(u) = \varrho \cdot \frac{2u - 1}{\text{err}(u)} \quad , \quad \psi^{Q^{-1}}(z) = \frac{\varrho + \text{H}(-z)}{2\varrho + |z|} \quad (12)$$

its convex surrogate is

$$F^Q(z) = -\varrho \cdot \log \varepsilon - \varrho \cdot \log\left(2 + \frac{|z|}{\varrho}\right) + \text{H}(z). \quad (13)$$

The proof (SM, Section 2), includes a technical discussion explaining the reason of the ε parameter for properness. We insist however on the fact that ε plays no role in the minimisation of the loss or its convex surrogate. Figure 2 plots the matching prediction for the Q-loss. We remark that it does not suffer the same constraints as for the square loss. Unlike the square loss, $F^Q(.)$ is strictly decreasing over $\mathbb{R}$ and it is 1-Lipschitz (direct consequence of (7), (8)).

We now define following Collins et al. (2000) the mirror update for the Q-loss:

$$z \diamond u \doteq \psi^{Q^{-1}}\left(-z + \psi^Q(u)\right), \quad (14)$$

where $z \in \mathbb{R}$, $u \in (0, 1)$. The following Lemma gives a concrete idea of how simple the mirror update is using the Q-loss when inputs are rational in the context of our boosting application (that is, when $z \propto \varrho$).

Algorithm 1 RATBOOST

Input sample $\mathcal{S} = \{(\boldsymbol{x}_i, y_i), i = 1, 2, ..., m\}$, number of iterations T, $\varrho \in \mathbb{N}_*$, $a \in \mathbb{Q}_{+*}$;
Step 1 : let $\boldsymbol{\alpha} \leftarrow \mathbf{0}$;
Step 2 : let $w_i = 1/2, \forall i = 1, 2, ..., m$; // initial weights
Step 3 : **for** $t = 1, 2, ..., T$
Step 3.1 : let $j \leftarrow \text{WL}(\mathcal{S}, \boldsymbol{w})$
Step 3.2 : let $\eta(j) \leftarrow (1/m) \cdot \sum_i w_i y_i h_j(\boldsymbol{x}_i)$
Step 3.3 : let $\delta_j \leftarrow a\eta(j)$
Step 3.4 : **for** $i = 1, 2, ..., m$, let

$$w_i \quad \leftarrow \quad \delta_j y_i h_j(\boldsymbol{x}_i) \diamond w_i \ ; \tag{21}$$

Step 3.5 : let $\alpha_j \leftarrow \alpha_j + \delta_j$ // update of α_j
Return $H_T = \sum_j \alpha_j h_j$.

Lemma 6 *Suppose $u = n_u/d_u$ (with $n_u \in \mathbb{N}, d_u \in \mathbb{N}_*, n_u \leq d_u$), $z = \varrho \cdot n_z/d_z$ (with $n_z \in \mathbb{Z}, d_z \in \mathbb{N}_*$ wlog). Denote for short*

$$a_1 \doteq (n_u \wedge (d_u - n_u)) \cdot d_z \quad (\in \mathbb{N}), \tag{15}$$
$$a_2 \doteq (n_u \wedge (d_u - n_u)) \cdot n_z \quad (\in \mathbb{Z}), \tag{16}$$
$$a_3 \doteq d_u \cdot d_z \quad (\in \mathbb{N}_*), \tag{17}$$
$$a_4 \doteq 2 \cdot (d_u - 2n_u) \cdot d_z \quad (\in \mathbb{Z}). \tag{18}$$

Then,

$$z \diamond u = \begin{cases} \frac{a_1}{a_2 + a_3 - \text{H}(a_4)} & \text{if} \quad a_2 + (a_4/2) \geq 0 \\ 1 - \frac{a_1}{-a_2 + a_3 - \text{H}(-a_4)} & \text{otherwise} \end{cases} \tag{19}$$

(proof in SM, Section 4) In the general case, the mirror update takes the simplified form:

$$z \diamond u = \frac{\varrho \cdot \text{err}(u) + H\left(z \cdot \text{err}(u) + \varrho \cdot (1 - 2u)\right)}{2\varrho \cdot \text{err}(u) + |z \cdot \text{err}(u) + \varrho \cdot (1 - 2u)|}. \tag{20}$$

5. Boosting ensembles

We train linear combinations of weak classifiers to minimize the Q-loss. Algorithm RATBOOST is described in Algorithm 1. As we shall see, RATBOOST is a formal boosting algorithm. As such, it relies on calls to a *weak learner*, $\text{WL}(\mathcal{S}, \boldsymbol{w})$, which takes as input sample $\mathcal{S}$ and weights $\boldsymbol{w}$ over the examples, and returns a weak hypothesis. To simplifiy notation, the weak learner returns the index j of the weak hypothesis in a set of possible choices. Quantity $\eta(j)$, the edge of the weak hypothesis, is used to compute a leveraging update δ_j for the classifier in the set. Step 3.4 is fundamental for boosting: it is the update of weights $\boldsymbol{w}$ following the mirror update of the Q-loss, which defines a "memory" of previous boosting iterations. A minor difference emerges with variants (Schapire & Singer, 1999; Nock & Nielsen, 2007): weights are in $[0, 1]$ but not normalized. A major difference with previous boosting algorithms is however immediate using Lemma 6: suppose without loss of generality that the input of RATBOOST is encoded in $\mathbb{Q}$ (training sample $\mathcal{S}$ and parameters a, ϱ) and the weak hypotheses of $\text{WL}(\mathcal{S}, \boldsymbol{w})$ have range in $\mathbb{Q}$. Then,

RATBOOST requires only integer functions $+, -, /, \times, |.|$

and the boosted classifier H_T *also* has range in $\mathbb{Q}$. The absolute value function can equivalently be replaced by $\text{H}(.)$ or sign, still with integer domain.

RATBOOST is a boosting algorithm We now investigate the boosting abilities of RATBOOST, and for this objective, we make the following assumption:

(M) $\exists$ constant $M > 0$ s. t. $\sqrt{\mathbb{E}_{i \sim [m]}\left[h_j^2(\boldsymbol{x}_i)\right]} \leq M, \forall j$.

(M) bounds the second moment of $h_.$, and would be implied if for example the range of $h_.$ were bounded. Hereafter, we shall re-index the edge by the iteration number (η_t) instead of the weak classifier index, for readability.

Theorem 7 *Consider the following in* RATBOOST*:*

(i) assumption (M) holds;
(ii) pick $\pi \in [0, 1)$, $a \in \frac{2\varrho}{M^2} \cdot [1 - \pi, 1 + \pi]$ (Step 3.3);

Then for any $z^* \geq 0$, if the sequence of edges $\{\eta_t\}_t$ satisfies

$$\frac{1}{M^2} \cdot \sum_{t=1}^{T} \eta_t^2 \quad \geq \quad \frac{F^Q(0) - F^Q(z^*)}{(1 - \pi^2)\varrho}, \tag{22}$$

then $\mathbb{E}_D\left[F^Q(y_i H_T(\boldsymbol{x}_i))\right] \leq F^Q(z^*)$.

(proof in SM, Section 4) Suppose we make the following *weak learning assumption*: there exists a small constant $\gamma > 0$ such that $|\eta_t| \geq \gamma, \forall t$ (Nock & Nielsen, 2008; Schapire & Singer, 1999). Then we have $\mathbb{E}_D\left[F^Q(y_i H_T(\boldsymbol{x}_i))\right] \leq F^Q(z^*)$ as soon as the number of iterations T satisfies

$$T \quad \geq \quad \frac{M^2(F^Q(0) - F^Q(z^*))}{(1 - \pi^2)\gamma^2\varrho}, \tag{23}$$

which is indeed boosting compliant (Schapire et al., 1998, Section 2), (Nock & Nielsen, 2008). We remark the freedom in choosing parameter a with which we can enforce $a \in \mathbb{Q}$.

Quantizing RATBOOST We now provide a sufficient condition for a quantization of the weights in RATBOOST to keep the boosting convergence property. Importantly, the result does not rely on a specific form of the quantization but on a dominance of the quantization residuals with respect to the edge. Since the available edges $\eta_.$ should intuitively vanish with the number of boosting iterations as we reach the global optimum, it means that quantization shall be efficient

until a certain iteration, after which we cannot guarantee boosting convergence. Since real numbers encoded in a computer are finite and inevitably quantized (Widrow & Kollár, 2008), this message extends to RATBOOST as well, with less stringent conditions on convergence though.

Hereafter, we let $\tilde{\boldsymbol{w}}$ denote a vector of quantized weights. As we shall see later, values in $\tilde{\boldsymbol{w}}$ can also be progressively learned. Since the initial weight value in RATBOOST is $1/2$, it is important that at least the initial $\tilde{\boldsymbol{w}}$ contains this value. When the quantization is regular, it is also important that we can encode u and $1-u$ for all available us, so one should pick $\dim(\tilde{\boldsymbol{w}})$ odd. We then define Q-RATBOOST as RATBOOST in which we replace weight update in Step 3.4 by the following step:

Step 3.4 : **for** $i = 1, 2, ..., m$, let

$$\tilde{w}_i \quad \leftarrow \quad \arg\min_{w^* \in \tilde{\boldsymbol{w}}} |\tilde{\delta}_j y_i h_j(\boldsymbol{x}_i) \diamond \tilde{w}_i - w^*| \;; \quad (24)$$

The tilda notation additionally renames parameters whose value may depend on the quantization of $\tilde{\boldsymbol{w}}$ (we also define $\tilde{\eta}$ accordingly). Let

$$\kappa_j \quad \doteq \quad \left| \frac{\sum_i (\tilde{w}_{ji} - w_{ji}) y_i h_j(\boldsymbol{x}_i)}{m} \right| \quad (25)$$

denote the error due to quantization in the weight update, where $w_{\cdot}$ denotes the output of the weight update as in (21) and Lemma 6. If there were no weight quantization, κ_j would be zero. As we did for $\eta_{\cdot}$, we reindex for the sake of our formal results $\kappa_{\cdot}$ by the iteration number (κ_t) and consider the following assumption:

(Q) there exists $\zeta > 0$ such that the quantization error is dominated by the edge: $\kappa_t \leq (1-\zeta) \cdot |\tilde{\eta}_t|$, for any t.

Before commenting on this assumption, let us show that (Q) still allows for boosting.

Theorem 8 *Consider the following in* Q-RATBOOST*:*

(i) assumptions (M) and (Q) hold;

(ii) pick $\pi \in [0, \zeta)$*;* $a \in \frac{2\varrho}{M^2} \cdot [\zeta - \pi, \zeta + \pi]$ *(Step 3.3);*

Then for any $z^* \geq 0$*, if the sequence of edges* $\{\eta_t\}_t$ *satisfies*

$$\frac{1}{M^2} \cdot \sum_{t=1}^{T} \eta_t^2 \quad \geq \quad \frac{F^Q(0) - F^Q(z^*)}{(\zeta^2 - \pi^2)\varrho}, \quad (26)$$

then $\mathbb{E}_D \left[F^Q(y_i H_T(\boldsymbol{x}_i)) \right] \leq F^Q(z^*)$.

(proof in SM, Section 5) If we make the weak learning assumption on quantized edges, $|\tilde{\eta}_t| \geq \gamma, \forall t$ for some $\gamma > 0$, then we have $\mathbb{E}_D \left[F^Q(y_i H_T(\boldsymbol{x}_i)) \right] \leq F^Q(z^*)$ as soon as the number of iterations T satisfies

$$T \quad \geq \quad \frac{M^2 (F^Q(0) - F^Q(z^*))}{(\zeta^2 - \pi^2)\gamma^2 \varrho}. \quad (27)$$

What is interesting in the context of boosting is that the weak learning assumption guarantees that (Q) can be satisfied for a moderate size encoding of a regular quantization, as explained in the following Lemma (proof immediate).

Lemma 9 *Suppose the weak assumption holds and (M) is replaced by the condition* $|h_j| \leq M, \forall j$*. Then (Q) holds if* $\tilde{\boldsymbol{w}}$ *is a regular quantization and*

$$\dim(\tilde{\boldsymbol{w}}) \quad \geq \quad 1 + \frac{M}{2(1-\zeta)\gamma}. \quad (28)$$

Hence, in theory, quantized boosting requires no more assumption than the weak learning assumption. To get an idea of the space required to store or communicate $\tilde{\boldsymbol{w}}$, suppose $\zeta = 1/2, \pi = 0$, which implies that the iteration bound on T is four times that for boosting without quantization. For a regular quantization, the amount of bits $\tilde{\mathrm{b}}(\mathcal{S})$ to store (or communicate) $\tilde{\boldsymbol{w}}$ and the index of the quantized weight for each example in the training sample is therefore

$$\tilde{\mathrm{b}}(\mathcal{S}) \quad = \quad O\left(\left(m + \frac{M}{\gamma} \right) \cdot \log\left(\frac{M}{\gamma} \right) \right), \quad (29)$$

whereas without quantization, the corresponding amount, $\mathrm{b}(\mathcal{S})$, would rather satisfy

$$\mathrm{b}(\mathcal{S}) \quad = \quad O\left(mk\right), \quad (30)$$

provided we can make a *lossless* encoding of each weight in a k-bits data structure for a fair comparison, which is everything but simple. Even in this case, we see that quantisation can be fairly advantageous if γ is not too small, such as for example $\gamma \gg 1/k$. For a 64 bits encoding of weights in RATBOOST, this could be achieved with the weak learner learning not too weak classifiers. We shall see how this is theoretically possible with decision trees in Section 7.

6. Experiments

We implemented the following variants of RATBOOST:
▷ RATBOOST refers to Algorithm 1 ran with unconstrained inputs and the weight update as in (20). Numbers are encoded as `double` precision;
▷ RATBOOSTQ$_b$ is Q-RATBOOST with a deterministic regular quantization of weights on b bits, which brings a $2^b - 1$ regular quantization of $[0, 1]$, used as in (24).
▷ RATBOOSTS$_b$ is RATBOOSTQ$_b$ in which we replace the deterministic quantization by a stochastic one following the stochastic rounding of Gupta et al. (2015).

		AdaBoost	AdaBoost$_{\mathbb{R}}$	RatBoost	RatBoostE	RatBoostQ$_b$, $b=$		RatBoostS$_b$, $b=$		RatBoostA$_b$, $b=$	
						2	6	2	6	2	6
F	3	38.00±10.33	37.00±9.49	40.00±9.43	40.00±11.55	47.00±14.94	42.00±11.35	38.00±7.89	52.00±16.19	41.00±7.38	39.00±8.76
H	3	25.53±8.79	25.53±8.79	25.85±8.32	26.81±9.71	25.84±9.83	26.81±9.71	25.84±9.34	26.17±10.01	25.52±9.65	25.84±8.31
T	4	38.78±6.86	39.05±6.68	38.92±7.15	34.91±7.25	39.18±7.02	40.66±7.98	30.24±8.27	38.24±6.92	33.59±8.06	35.71±5.39
B	4	2.70±1.62	2.63±1.55	2.99±1.70	4.89±1.89	15.46±2.59	13.93±2.97	7.73±3.12	3.43±1.33	12.54±3.22	3.57±2.16
BW	4	2.86±2.78	2.86±2.78	3.29±2.52	3.14±2.59	10.02±3.83	3.58±2.72	2.86±2.13	3.00±2.65	4.29±3.09	3.00±2.37
I	5	11.39±4.01	11.11±3.91	11.68±3.92	12.54±5.26	25.37±5.82	13.69±4.25	12.83±3.64	13.40±5.41	14.53±4.94	13.40±3.60
S	5	20.67±7.12	20.67±7.12	21.64±6.47	25.48±9.88	30.69±12.30	27.38±9.72	24.02±8.71	26.07±9.29	23.10±9.62	24.05±11.76
Y	5	48.18±4.43	48.18±4.43	48.59±4.59	34.04±7.09	48.52±4.00	48.45±4.60	49.33±3.83	46.77±3.67	48.79±2.64	49.33±3.97
WR	5	26.14±3.02	26.14±3.15	25.45±3.70	26.27±3.18	30.77±3.48	27.02±4.23	27.08±3.36	26.89±4.08	27.64±3.45	25.96±3.39
Ca	5	41.63±4.62	41.58±4.55	39.23±4.46	37.91±2.88	45.86±2.04	42.47±2.80	42.06±4.33	42.00±2.27	38.05±3.89	37.11±3.54
CCS	5	40.00±4.62	39.90±4.70	40.90±3.31	39.90±4.56	57.90±4.12	42.10±5.95	39.90±3.87	36.40±4.40	42.60±4.58	40.90±4.33
Ab	5	21.64±1.81	21.62±1.86	21.35±1.60	22.10±1.41	24.18±1.55	24.40±1.42	22.86±1.54	21.52±1.46	24.28±1.40	21.74±1.45
Q	5	22.47±6.54	22.37±6.50	19.81±5.14	20.48±5.55	31.47±4.82	22.65±4.44	22.47±5.49	22.37±5.72	24.55±4.60	21.24±5.74
WW	5	30.36±2.18	30.32±2.09	29.77±1.95	29.64±2.03	35.87±2.05	31.69±1.68	31.46±2.12	31.58±2.23	31.30±2.30	29.75±1.23
P	5	19.26±1.91	19.24±1.84	6.01±1.18	7.80±1.45	35.61±1.93	21.14±1.77	11.04±2.20	14.80±2.04	22.35±1.68	7.22±1.07
Mi	5	4.07±2.15	3.89±2.04	4.44±2.30	7.41±3.55	26.11±4.32	10.09±3.77	11.94±3.61	8.33±2.90	13.70±2.72	7.31±2.97
H+n	6	41.91±5.96	41.91±5.96	35.15±5.32	39.93±5.56	49.25±4.85	44.06±6.79	45.72±6.26	40.18±5.64	42.99±5.54	28.96±9.20
H+nn	6	41.99±5.45	41.99±5.45	32.91±5.07	37.95±4.98	48.76±4.78	41.99±8.40	42.58±6.41	37.47±5.10	41.66±4.26	19.63±8.76
Ft	6	12.23±0.93	12.39±0.90	12.33±0.85	13.56±1.07	33.78±1.62	13.74±0.75	17.13±1.13	12.56±0.82	21.12±1.42	12.57±0.90
Ma	6	21.00±1.00	21.01±0.93	20.91±0.97	20.94±0.98	26.41±0.97	21.11±0.88	20.95±0.93	20.94±0.96	21.45±0.88	21.01±0.96
E	6	45.55±1.48	45.55±1.49	43.48±1.36	42.92±0.81	47.26±1.43	44.07±1.45	45.63±1.71	44.51±1.80	44.83±1.50	42.25±0.72
Sk	6	9.62±0.22	10.18±0.29	10.74±0.21	9.65±0.23	33.97±0.29	9.87±0.23	9.74±0.24	9.61±0.23	9.62±0.23	6.77±0.34
Mu	7	23.36±1.19	23.28±1.24	19.48±1.12	22.26±1.19	46.07±5.17	26.20±1.46	32.92±2.97	28.13±2.28	28.54±2.03	24.72±1.04
Ha	7	1.94±0.23	3.11±0.31	3.11±0.33	1.76±0.25	9.73±0.44	2.28±0.20	1.80±0.32	2.72±0.19	2.29±0.23	1.66±0.17
Tw	8	7.45±0.08	7.45±0.08	4.42±0.11	4.72±0.10	6.63±0.07	5.65±0.14	4.34±0.11	5.07±0.11	5.21±0.19	4.90±0.15

Table 1: Test errors on UCI domains (excerpt). The single digit after the ID is #digits of md ($\propto \log$ (storing size)).

▷ RatBoostA$_b$ is RatBoostQ$_q$ in which we replace the regular quantization by an adaptive one learned using $(2^b - 1)$-means, updated at each boosting iteration. Note that the optimal solution of k-means is rational if inputs are (Banerjee et al., 2004), and there is an optimal polynomial algorithm for the 1D case (Nielsen & Nock, 2014; Zhang et al., 2017) – yet with a costly $O(m^2)$ time *and* space complexity: since we renew quantization at *each* iteration, we have instead opted for a fast Forgy k-means, which experimentally approximately converges in few iterations.

▷ RatBoostE is RatBoost ran with all numbers (including inputs) represented as rational fractions with `long` numerators and denominators, weight update as in Lemma 6, operations restricted to integer $+, -, \times, /, |.|$, and a deterministic quantization as in RatBoostQ$_q$ with $q = 11$ to manage overflows. When quantization is not used and inputs are rational (including training set and weak learning's classifiers *outputs*), RatBoostE implements the *lossless* solution to the iterative minimization.

In all algorithms, we fix $\varrho = 1$, estimate M for assumption (M) from the training sample and then compute a as in Theorem 7 with $\pi = 0$. Experiments are carried out with 10-folds stratified cross validation on 25 UCI domains (Blake et al., 1998). Contenders are AdaBoost (Schapire & Singer, 1999) and AdaBoost$_{\mathbb{R}}$ (Nock & Nielsen, 2007) (both algorithms have in common that they do not optimize a proper canonical loss). Table 1 gives results for a subset of the values for b. The Supplement (Section 7) details domains, per-domain results and provides a complete version of Table 1. The following conclusions can be drawn: RatBoostE competes with all other boosting algorithms, all the more on the biggest domains, with some very significant improvement observed (P). Quantization clearly brings benefits, RatBoostA$_.$ being a clear winner, with just 2-bit indexes allowing to compete with the best algorithms in several cases (Ca, E, Sk, Tw). Stochastic quantization leads to a much more erratic behaviour. Attempts to (adaptively) quantize AdaBoost for comparisons have met with mixed results against non quantized AdaBoost, and were clearly defeated by RatBoostA$_.$ on the bigger domains.

7. More and better boosting with the Q-loss

The state of the art decision tree (DT) induction algorithms historically consist of two phases: the induction of a large tree which is then pruned for good generalization (Breiman et al., 1984). Pruning is a statistical procedure out of the scope of our paper, but the top-down induction is more closely related: the state of the art indeed minimizes the pointwise Bayes risk of a given proper canonical loss, which can be the Gini entropy for CART (Breiman et al., 1984), the binary entropy for C4.5/C5 (Quinlan, 1993), Matsushita's criterion for the optimal tree algorithm of Dietterich et al. (1996); Kearns & Mansour (1996), etc. . Let H be a DT with a set Λ of leaves. For any leaf $\lambda \in \Lambda$, $q(\lambda)$ denotes the *relative* proportion of positive examples in that leaf and $\omega(\lambda)$ is the proportion of examples coming to that leaf (thus,

$\sum_\lambda \omega(\lambda) = 1$). Then the top-down induction repeatedly changes leaves by subtrees (starting from the tree being a single root leaf node) by minimizing the *tree*'s Bayes risk:

$$\underline{L}(H) \doteq \sum_{\lambda \in \Lambda : p(\lambda) \neq 0,1} \omega(\lambda) \underline{L}(q(\lambda)). \quad (31)$$

In the final tree, one can attribute class probability estimates at each leaf (Quinlan, 1996), but a convenient (and in fact equivalent) real-valued prediction at leaf λ is $\psi(q(\lambda))$ (Schapire & Singer, 1999), ψ being the canonical link of the loss. Notice the relevance of such trees in our context as

> the output of the tree learned using the Q-loss is *always* in $\mathbb{Q}$ if ϱ is also in $\mathbb{Q}$ (Theorem 5).

Also, we do not use *pure* leaves ($p(\lambda) \neq 0, 1$) in (31) without loss of generality as $\underline{L}(0) = \underline{L}(1) = 0$ for all the state of the art splitting criteria. In the context of DT, the weak learning assumption (WLA) of boosting can be restated as folllows: let $\boldsymbol{s}_\lambda \in \{0,1\}^m$ denote the predicate storing in index i whether training example i reaches λ in H (since each training example reaches one leaf, we have $\sum_\lambda \boldsymbol{s}_\lambda = \mathbf{1}$), $g : \mathbb{R}^d \to \{-1, 1\}$ denote a potential test put to split leaf λ (H is binary wlog) and $\eta(g, \boldsymbol{s}) \doteq (1/N_\lambda) \cdot \sum_{i \in \mathcal{S}_\lambda} w_i s_{\lambda i} y_i g(\boldsymbol{x}_i)$ with $N_\lambda \doteq \sum_{i \in \mathcal{S}_\lambda} w_i s_{\lambda i}$ a normalizing constant for weights. Then the WLA of Kearns & Mansour (1996) is equivalent to: there exists a small constant $\gamma > 0$ such that split g at leaf λ satisfies $\eta(g, \boldsymbol{s}) \geq 2\gamma$, which has a similar flavour to our WLA in § 5.

Since $\underline{L}^Q(u)$ takes values in $\varrho \cdot [-\infty, \log(1/(2\varepsilon))]$ and a DT partitions finite data in discrete subsets, we can consider wlog that ε is sufficiently small so that $\underline{L}^Q(p) > 0$ where $p = \boldsymbol{s}^\top \boldsymbol{w} / \mathbf{1}^\top \boldsymbol{w}, \forall \boldsymbol{s} \in \{0,1\}^m \backslash \{0^m, 1^m\}$. This is just a technical assumption for the result below to be meaningful and allow fair comparisons with (Kearns & Mansour, 1996). We also consider that no leaf in H is pure (with examples of just one class), a very reasonable assumption since (i) from a formal standpoint it does not endanger the WLA, and (ii) in practice DTs are pruned for good generalization so we end up with leaves indeed not pure with overwhelming probability (see for example the near-optimal pruning of Kearns & Mansour (1998)). The question we now address is: what is the minimal number of iterations T of the leaf-growing procedure to guarantee that the final tree H_T (thus with $T + 1$ leaves) will have $\underline{L}^Q(H_T) \leq \rho \cdot \underline{L}^Q(q(\mathcal{S}))$, where we fix $\rho \in (0, 1]$ and $q(\mathcal{S})$ is the proportion of positive examples in $\mathcal{S}$. The following Theorem answers this question. We let $v(\varepsilon, \varrho) \doteq 4\varrho / \log(1/(2\varepsilon)) > 0$.

Theorem 10 *Suppose WLA holds. Then for any $\rho \in (0, 1]$, we are guaranteed that $\underline{L}^Q(H_T) \leq \rho \cdot \underline{L}^Q(q(\mathcal{S}))$ if*

$$T \geq \left(\frac{1}{\rho}\right)^{\frac{v(\varepsilon,\varrho)}{\gamma^2}}. \quad (32)$$

(SM, Section 6) Remarkably, this convergence rate is optimal and guarantees rates of the same order as the best DT splitting criterion of Kearns & Mansour (1996, Section 4). In particular, we significantly improve on both CART and C.5 guarantees (Kearns & Mansour, 1996, Theorem 1). The strong dependence on the Q-loss parameters ε, ϱ is not surprising as both influence the range and curvature (hence, concavity) of Bayes risk (Kearns & Mansour, 1996).

8. Discussion and Conclusion

Disregarding the boosting and convergence results, one can wonder whether simple transformations of an existing loss would grant the same (a-d) properties as for the Q-loss (See Introduction), in addition to the boosting rates we get. While popular losses (log, exp, square) fail at at least one of the properties, the existing statistical theory includes the toolbox that could allow the adaptation of existing losses, in particular by changing the link function to create so-called *proper composite* losses (Reid & Williamson, 2010). It is however unclear how to get all the statistical, arithmetic and boosting properties we grant with the Q-loss, but certainly worth investigating to optimize the *arithmetic* of boosting.

Theory and experiments suggest that this may be highly desirable: while our formal results give only a sufficient condition for weight quantization to allow boosting, our results suggest that a weight encoding optimized for boosting is in fact the key to boost better (and for a longer time) than with standard IEEE machine encodings. The recent literature has shown how this problem can benefit deep learning by reducing training time, energy and model sizes. Such concerns are far from being recent: Dasgupta et al. (1990) explored bitsize compression convergence of LMS almost thirty years before this was done for SGD (Bernstein et al., 2018), and the complexity of evaluating even simple-looking updates such as the Q-loss mirror update have been the object of studies for decades (Strassen, 1972). All our results point to the need to explore this problem further for the elegant machinery of boosting as it was primarily designed (Kearns, 1988). The ability to compute the lossless solution of the boosting problem has an experimental impact, since quantization can become the only source of approximation – yet it may necessitate to optimize integer `encoding` to eventually delay further the use of `quantization`.

Our resuls also show that such considerations might prove worthy in the longstanding debate on several remarkable abilities of boosting (Mease & Wyner, 2008), which has been so far mainly been driven by statistical or convexity arguments. Numerical handling issues, even when long known, have never really made it to qualify arguments in the debate, even less so weight arithmetic / encoding. The perspective that a reduced set of `operations` may act as a regularization process is interesting for generalization.

References

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. QSGD: communication-efficient SGD via gradient quantization and encoding. In *NIPS*30*, pp. 1707–1718, 2017.

Banerjee, A., Merugu, S., Dhillon, I., and Ghosh, J. Clustering with bregman divergences. In 4^{th} *SIAM SDM*, pp. 234–245, 2004.

Banner, R., Hubara, I., Hoffer, E., and Soudry, D. Scalable methods for 8-bit training of neural networks. In *NeurIPS*31*, pp. 5151–5159, 2018.

Bernstein, J., Wang, Y., Azizzadenesheli, K., and Anandkumar, A. SIGNSGD: compressed optimisation for non-convex problems. In *35th ICML*, pp. 559–568, 2018.

Blake, C. L., Keogh, E., and Merz, C. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Boser, B. E., Guyon, I., and Vapnik, V. A training algorithm for optimal margin classifiers. In *Proc. of the 5th International Conference on Computational Learning Theory*, pp. 144–152, 1992.

Bottou, L. and Bousquet, O. The tradeoffs of large scale learning. In *NIPS*20*, pp. 161–168, 2007.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge University Press, 2004.

Breiman, L., Freidman, J. H., Olshen, R. A., and Stone, C. J. *Classification and regression trees*. Wadsworth, 1984.

Buja, A., Stuetzle, W., and Shen, Y. Loss functions for binary class probability estimation and classification: structure and applications, 2005. Technical Report, University of Pennsylvania.

Collins, M., Schapire, R., and Singer, Y. Logistic regression, adaboost and Bregman distances. In *Proc. of the 13th International Conference on Computational Learning Theory*, pp. 158–169, 2000.

Dasgupta, S., Johnson, C.-R., and Baksho, A.-M. Sign-sign LMS convergence with independent stochastic inputs. *IEEE Trans. IT*, 36:197–201, 1990.

Dietterich, T. G., Kearns, M. J., and Mansour, Y. Applying the weak learning framework to understand and improve C4.5. In *13th ICML*, pp. 96–104, 1996.

Drumond, M., Lin, T., Jaggi, M., and Falsafi, B. Training dnns with hybrid block floating point. In *NeurIPS*31*, pp. 451–461, 2018.

Freund, Y. and Schapire, R. E. A Decision-Theoretic generalization of on-line learning and an application to Boosting. *J. Comp. Syst. Sc.*, 55:119–139, 1997.

Friedman, J., Hastie, T., and Tibshirani, R. Additive Logistic Regression : a Statistical View of Boosting. *Ann. of Stat.*, 28:337–374, 2000.

Goldwasser, S. Machine learning and cryptography: Challenges and opportunities. NeurIPS'18 workshop on Privacy Preserving Machine Learning, 2018.

Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. Deep learning with limited numerical precision. In *32nd ICML*, pp. 1737–1746, 2015.

Hardy, S., Henecka, W., Ivey-Law, H., Nock, R., Patrini, G., Smith, G., and Thorne, B. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *CoRR*, abs/1711.10677, 2017.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. In *NIPS*29*, pp. 4107–4115, 2016.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A.-G., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proc. of the 31th IEEE CVPR*, pp. 2704–2713, 2018.

Kearns, M. Thoughts on hypothesis boosting, 1988. ML class project.

Kearns, M. and Mansour, Y. On the boosting ability of top-down decision tree learning algorithms. In *Proc. of the 28th ACM STOC*, pp. 459–468, 1996.

Kearns, M. J. and Mansour, Y. A Fast, Bottom-up Decision Tree Pruning algorithm with Near-Optimal generalization. In *Proc. of the 15th International Conference on Machine Learning*, pp. 269–277, 1998.

Kohavi, R. Improving accuracy by voting classification algorithms: Boosting, bagging, and variants. In *Workshop on Computation-Intensive Machine Learning Techniques*, 1998.

Lazarevic, A. and Obradovic, Z. Boosting algorithms for parallel and distributed learning. *Distributed and Parallel Databases*, 11:203–229, 2002.

Lin, D.-D., Talathi, S.-S., and Annapureddy, V.-S. Fixed point quantization of deep convolutional networks. In *33rd ICML*, pp. 2849–2858, 2016.

Mease, D. and Wyner, A. Evidence contrary to the statistical view of boosting (with discussion). *JMLR*, 9:131–201, 2008.

Narang, S., Diamos, G., Elsen, E., Micikevicius, P., Alben, J., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., and Wu, H. Mixed precision training. In *ICLR'18*, 2018.

Nielsen, F. and Nock, R. Optimal interval clustering: Application to Bregman clustering and statistical mixture learning. *IEEE Signal processing letters*, 21:1289–1292, 2014.

Nock, R. and Nielsen, F. A $\mathbb{R}$eal Generalization of discrete AdaBoost. *Artificial Intelligence*, 171:25–41, 2007.

Nock, R. and Nielsen, F. On the efficient minimization of classification-calibrated surrogates. In *NIPS*21*, pp. 1201–1208, 2008.

Qin, D., Chen, X., Guillaumin, M., and Van Gool, L. Quantized kernel learning for feature matching. In *NIPS*27*, 2014.

Quinlan, J. R. *C4.5 : programs for machine learning*. Morgan Kaufmann, 1993.

Quinlan, J. R. Bagging, boosting and c4.5. In *Proc. of the 13^{th} National Conference on Artificial Intelligence*, pp. 725–730, 1996.

Reid, M.-D. and Williamson, R.-C. Composite binary losses. *JMLR*, 11:2387–2422, 2010.

Robbins, H. and Monro, S. A stochastic approximation method. *Ann. Math. Stat.*, 22:400–407, 1951.

Rumelhart, D.-E., Hinton, G.-E., and Williams, R.-J. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

Sakr, C., Kim, Y., and Shanbhag, N.-R. Analytical guarantees on numerical precision of deep neural networks. In *34^{th} ICML*, 2017a.

Sakr, C., Patil, A.-D., Zhang, S., Kim, Y., and Shanbhag, N.-R. Minimum precision requirements for the SVM-SGD learning algorithm. In *42^{nd} IEEE ICASSP*, pp. 1138–1142, 2017b.

Schapire, R. E. and Singer, Y. Improved boosting algorithms using confidence-rated predictions. *MLJ*, 37:297–336, 1999.

Schapire, R. E., Freund, Y., Bartlett, P., and Lee, W. S. Boosting the margin : a new explanation for the effectiveness of voting methods. *Annals of statistics*, 26:1651–1686, 1998.

Slawski, M. and Li, P. b-bit marginal regression. In *NIPS*28*, 2015.

Strassen, V. Evaluation of rational functions. In Milner, R.-E. and Thatcher, J.-W. (eds.), *Complexity of computer computations*, pp. 1–10. Plenum Press, 1972.

Vapnik, V. *Statistical Learning Theory*. John Wiley, 1998.

Widrow, B. and Kollár, I. *Quantization Noise*. Cambridge University Press, 2008.

Zhang, H., Li, J., Kara, K., Alistarh, D., Liu, J., and Zhang, C. ZipML: Training linear models with end-to-end low precision, and a little bit of deep learning. In *ICML'17*, pp. 4035–4043, 2017.