Gentle Nearest Neighbors Boosting over Proper Scoring Rules

Richard Nock, Wafa Bel Haj Ali, Roberto D’Ambrosio, Frank Nielsen (Senior Member, IEEE), Michel Barlaud (Fellow, IEEE)

Abstract—Tailoring nearest neighbors algorithms to boosting is an important problem. Recent papers study an approach, \textit{UNN}, which provably minimizes particular convex surrogates under weak assumptions. However, numerical issues make it necessary to experimentally tweak parts of the UNN algorithm, at the possible expense of the algorithm’s convergence and performance. In this paper, we propose a lightweight Newton-Raphson alternative optimizing proper scoring rules from a very broad set, and establish formal convergence rates under the boosting framework that compete with those known for UNN. To the best of our knowledge, no such boosting-compliant convergence rates were previously known in the popular Gentle Adaboost's lineage. We provide experiments on a dozen domains, including Caltech and SUN computer vision databases, comparing our approach to major families including support vector machines, (Ada)boosting and stochastic gradient descent. They support three major conclusions: (i) \textit{GNNB} significantly outperforms UNN, in terms of convergence rate and quality of the outputs, (ii) \textit{GNNB} performs on par with or better than computationally intensive large margin approaches, (iii) on large domains that rule out those latter approaches for computational reasons, \textit{GNNB} provides a simple and competitive contender to stochastic gradient descent. Experiments include a divide-and-conquer improvement of \textit{GNNB} exploiting the link with proper scoring rules optimization.

1 INTRODUCTION

Iterative approaches to learn classifiers have been playing a major role in machine learning and statistical learning for many decades. The most common high-level scheme consists in gradually combining from scratch classifiers obtained at each iteration, with the objective to minimize throughout iterations a convex differentiable risk called a surrogate risk, sometimes amended with a structural part based on data [1]. Unlike so-called greedy algorithms, that repeatedly perform fine-grained optimization steps [1], boosting algorithms rely on weak optimization stages much less demanding from the statistical and computational standpoints [2], [3], [4], [5]. In fact, the boosting theory involves at each iteration weak classifiers slightly different from pure random, \textit{but} requires that the final combination be probably as close as required from optimum, in polynomial time.

Nearest neighbors (NN) rules are a non-trivial field of choice for boosting algorithms [3], [4], as examples ideally play weak classifiers. In this case, we treat the boosting problem in its simplest form: the accurate leveraging of examples that vote among nearest neighbors. In particular, we compute nearest neighbors in the ambient space of data, \textit{i.e.} as described over their initial features. There have been other approaches to boost nearest neighbors by learning features with (Ada)boosting algorithms, prior to computing nearest neighbor rules on these new sets of features [6] (and references therein). No boosting results are known for these algorithms, and it is in fact not known whether they achieve convergence to the optimum of Adaboost’s exponential risk. A previous approach in our line of works is algorithm UNN (for “Universal Nearest Neighbors”), which brings boosting guarantees for merely all strictly convex differentiable surrogates relevant to classification [7], [4]. For a wide subset of surrogates, it yields simple and efficient estimators of posteriors [8].

There is, however, an analytical and computational bottleneck in UNN, as the leveraging coefficients are solutions to non-linear equations with no closed form expression in the general case. Boosting compliant approximations are possible, but in the context of NN rules, they are computationally far too expensive to be performed at each boosting iteration on large datasets. Computationally affordable coarse-grained approximations are also possible, that yield compelling experimental results, but it is not known if they always lie within the boosting regime [4].

In this paper, we propose a simple boosting compliant solution to this computational bottleneck. Our algorithm, \textit{GNNB} for “Gentle Nearest Neighbors Boosting”, performs adaptive Newton-Raphson steps to minimize any balanced convex surrogate [9] with guar-
From left to right: examples of balanced convex losses \( \psi_\phi \) (A, B, C, D; we let \( \ln \) denote the base-e logarithm, and \( \log_2(x) = \ln(x)/\ln(2) \)); permissible functions \( \phi \); value of \( \pi^* \) as defined in (41); expression of update \( \delta_j \) in (10) for \( \varepsilon = 1/2 \); expression of the weight update in (11) (See text for details).

| \( A \) \( (1 - x)^2 \) | \( \phi \) | \( -x(1-x) \) | \( \frac{1}{10} \) | \( \frac{n(c,j)}{2n_f} \) | \( w_i - \delta_j y_{ic} y_{jc} \) |
| \( B \) \( \log_2(1 + \exp(-x)) \) | \( x \ln x \) | \( + (1 - x) \ln(1-x) \) | \( \ln 2 \) | \( \frac{4(\ln(c,j))}{n_f} \) | \( \frac{w_i}{w_i \ln 2 + (1 - w_i) \ln 2 \times \exp(\delta_j y_{ic} y_{jc})} \) |
| \( C \) \( \log_2(1 + 2^{-x}) \) | \( x \log_2 x \) | \( + (1 - x) \log_2(1-x) \) | \( \ln(2) \) | \( \frac{4\eta(c,j)}{n_f} \) | \( \frac{w_i}{w_i + (1 - w_i) \times 2^\delta \eta y_{ic} y_{jc}} \) |
| \( D \) \( -x + \sqrt{1 + x^2} \) | \( -\sqrt{2(1-x)} \) | \( 1 \) | \( \frac{\eta(c,j)}{n_f} \) | \( 1 - \frac{1 - w_i + \sqrt{w_i(2 - w_i)} \delta_j y_{ic} y_{jc}}{\sqrt{w_i(2 - w_i)} \delta_j y_{ic} y_{jc}} \) |
| \( E \) \( \frac{1}{2}(\text{sign}(x) - 1) \) | \( -\min\{x, 1 - x\} \) | \( \text{N/A} \) |
means observation does not belong to the positive class, or similarly belongs to a “negative” class. In this case, a classifier \( h \) outputs a single real value.

More general than the problem of predicting labels is the problem of estimating posteriors \([10],[15]\): let \( p \equiv \hat{p}[y = 1|x] \) define for short the unknown true posterior for observation \( x \). The discrepancy between an estimator \( \hat{p} \) of \( p \) and \( p \) is measured by a loss \( \ell\). The interval \([0,1]\) in index recalls that its arguments are probabilities, and “\( || \)” means that it is not assumed to be symmetric. There are three requirements one can put on a loss to fit it to statistical requirements of the estimation task while making it suited to convenient algorithmic minimization.

The most important one, requirement \( R1 \), is fundamental in estimation, as it states that \( \ell\) defines a (strictly) proper scoring rule: \( 0 = \ell\) for any \( q \) and \( p \neq q\), and \( p \neq q\) for any permissible and satisfies (7). Since permissible is closed by linear combinations, function \( \phi + \phi_2 \) is also permissible and satisfies (7). Since \( H_{\psi}(x) = 1/[\phi(0) - \phi(1/2)] \times H_{\psi}(\nabla^{-1}_\psi(x)) \), assumption (7) implies:

\[
\min_{[0,1]} H_{\psi}(x) > 0 ;
\]

this is the case for all examples in Table 1. Otherwise, we may replace \( \phi \) by \( \phi + \phi_2 \) where \( \phi_2 \) is permissible and meets assumption (7). Since permissible is closed by linear combinations, function \( \phi + \phi_2 \) is also permissible and satisfies (7).

\[ H^*_\psi = \sup_{\mathbb{R}} H_{\psi}(x) \ll \infty \]

and in fact \( H^*_\psi = H_{\psi}(0) \) for all examples in Table 1, and is very small (Cf column \( \pi^* \), (41) and Section 4). The following Lemma states properties shown in [14].

**Lemma 1:** [14] For any permissible \( \phi \), the following properties hold true: \( \phi^*(x) = \phi^*(y - x) + y \forall x ; \nabla \psi(0) < 0 \), \( \psi(0) = 1 \), \( \im(\psi) \subseteq \mathbb{R}^+ \).

**2.3 Empirical risk and its minimization**

Lemma 1 makes that surrogate risk minimization may be used as an approximate primer to the minimization of the empirical risk, as the total surrogate risk (1) upperbounds the empirical (Hamming) risk [5]:

\[
\varepsilon_{\mathcal{S}}^H(\mathcal{H}) \geq \frac{1}{C} \sum_{c=1}^{C} \varepsilon_{\mathcal{S}}^{\psi^1}(h_c, c) \leq \frac{1}{\psi(0)} \varepsilon_{\mathcal{S}}^{\psi}(\mathcal{H}) ,
\]

where

\[
\varepsilon_{\mathcal{S}}^{\psi^1}(h_c, c) \equiv \frac{1}{m} \sum_{i=1}^{m} I[y_i h_c(x_i) < 0]
\]
Algorithm 1: Algorithm GENTLE NN BOOSTING, GNNB(S, φ, ε, k)

Input: S = {([x_i, y_i], i = 1, 2, ..., m, x_i ∈ O, y_i ∈ {−1, 1})}, permissible φ ∈ (0, 1), k ∈ N;
Let c, j = 0, ∀j = 1, 2, ..., m;
for c = 1, 2, ..., C do
Let w ← \frac{1}{2 \sqrt{0.75 + \varepsilon (1/2)}};

[I.1]//Choice of the example to leverage
Let j ← Win(S, w);

[I.2]//Computation of the gentle leveraging coefficient update, δ_j
Let
\eta(c,j) = \sum_{i ∼ S, k} w_i y_i c y_j ;
\delta_j = \frac{2(1 - ε)\eta(c,j)}{H_{c,j}^\varepsilon \ y_j}, with n_j = |\{i : j ∼ k i\}| ;

[I.3]//Weights update
∀i : j ∼ c, i, let
w_i ← \frac{1}{\nabla_{-1}^{-1}(-δ_j y_i c y_j + \nabla_{0}((\phi(0) - \phi(1/2)) w_i))} ;
\nabla_{0} = \phi(0) - \phi(1/2) ;
// we have w_i ∈ [0, (\phi(0) - \phi(1/2))^{-1}]

[I.4]//Leveraging coefficient update
Let α_jc ← α_jc + δ_j ;

Output: H(x) = \sum_{j ∼ k x} α_j ∩ y_j

Fig. 2. Weight update w′ computed as a function of w and g(c,j) = \eta(c,j)/n_j, when y_i c y_j = 1 and ε = 1/2 (see Table 1). The corresponding BCLS are the binary logistic loss (left) and Matsushita’s loss (right). The black grid depicts the plane of equation w′ = w.

is the usual empirical risk associated to class c. To quantify the performance of the best possible classifier, we respectively define:

\( (\varepsilon^S_{\phi})^c_c = \inf_h \varepsilon^S_{\phi} (h, c) \),
\( (\varepsilon^H_{\phi})^c_c = \inf_h \varepsilon^H_{\phi} (h, c) \),

as the respective Bayes surrogate risks and Bayes empirical risks for class c. Averaging these expressions following (1) and (12), we respectively define \((\varepsilon^S_{\phi})^s\) and \((\varepsilon^H_{\phi})^s\) as the optimal total surrogate risk and empirical (Hamming) risk on S. As a last remark, our minimization problems on the learning sample may be useful as well to minimize the true (surrogate) risks, that is, expectations of (1, 12) in generalization, according to some unknown distribution from which S is supposed i.i.d. sampled. We refer to [7] and the references therein for details, not needed here.

3 GENTLE BOOSTING FOR NN RULES

The nearest neighbors (NNs) rule belongs to the simplest classification algorithms [17]. It relies on a non-negative real-valued “distance” function. This function, defined on domain O, measures how much two observations differ from each other. It may not be a metric. We let j ∼ k x denote the assertion that example (x_j, y_j), or simply example j, belongs to the k NNs of observation x. We abbreviate j ∼ k x_i by j ∼ k i — and we say that example i belongs to the inverse neighborhood of example j. To classify an observation x ∈ O, the k-NN rule \(H\) over S computes the sum of class vectors of its nearest neighbors, that is:

\(H(x) = \sum_{j ∼ k x} \alpha_j ∩ y_j\), where \(\alpha\) is the Hadamard product. \(H\) predicts that x belongs to each class whose corresponding coordinate in the final vector is positive. A leveraged k-NN rule generalizes this to:

\(H(x) = \sum_{j ∼ k x} \alpha_j ∩ y_j\),

where \(\alpha_j \in \mathbb{R}^C\) is a leveraging vector for the classes in \(y_j\). Leveraging approaches to nearest neighbors are not new [18], yet to the best of our knowledge no convergence rates were known, at least until the algorithm UNN [4]. Algorithm 1 presents our gentle boosting algorithm for the nearest neighbor rules, GNNB. It differs with UNN on the key part of (16): the computation and update of the leveraging vectors. Instead of the repetitive solving of nonlinear equations — time consuming and with the risk, for approximations, of lying outside the boosting regime —, we prefer a simple scheme linear on the weighted edge \(\eta(c,j)\) (see Algorithm 1). The scheme of UNN [4] is nonlinear in this parameter. Our updates also depend on integer \(n_j\), the cardinality of the inverse neighborhood of example j, where |.| denotes the cardinality (see Algorithm 1). Table 1 gives the expressions of the weight update (11) for various choices of permissible \(\phi\) and the expression of \(\delta_j\) for the particular choice \(\varepsilon = 1/2\).

The ranges of values, used in Figure 2, are respectively \([-((\phi(0) - \phi(1/2))^{-1}), (\phi(0) - \phi(1/2))^{-1}]\) for \(g(c,j)\), and \([0, (\phi(0) - \phi(1/2))^{-1}]\) for w and \(w'\). The two plots, similar, exemplify two important remarks valid for any BCL. First, when classes match for example i and j, the weight of example i decreases iff \(\delta_j > 0\). This is a common behavior for boosting algorithms. Second, the regime of weight variations for extreme values of \(g(c,j)\) appear to be very important, despite the fact that leveraging update \(\delta_j\) is linear in the weighted edge. Thus, “gentle” updates do not prevent significant variations in weights.

4 PROPERTIES OF GNNB

4.1 GNNB is Newton-Raphson

Our first result establishes that GNNB performs Newton-Raphson updates to optimize its surrogate
risk, like Gentle Adaboost [2]. If we pick example \( i \) in the inverse neighborhood of example \( j \) to be updated for class \( c \), we have \( \partial \psi (y_{ic}, h_c (x_i)) / \partial j \neq - y_i \psi (y_{ic}, h_{jc}) \) and \( \partial^2 \psi (y_{ic}, h_c (x_i)) / \partial j^2 = H_{ic} (y_{ic}, h_i (x_i)) \), so that the Newton-Raphson update for \( \delta_j \) reads:

\[
\delta_j \leftarrow \rho \times \frac{\eta (c, j)}{\sum_{i:j \sim S, k} H_{ic} (y_{ic}, h_c (x_i))} \quad (17)
\]

for some small learning rate \( \rho \), typically with \( 0 < \rho \leq 1 \). Comparing with (10), we get the following result.

**Theorem 3:** GNNB uses adaptive Newton-Raphson steps to minimize the surrogate risk at hand, \( \varepsilon_S \), with adaptive learning rate \( \rho = \rho (c, j, \varepsilon) \):

\[
\rho (c, j, \varepsilon) = \frac{2 (1 - \varepsilon)}{\sum_{i:j \sim S, k} H_{ic} (y_{ic}, h_c (x_i))} \quad (18)
\]

Furthermore, \( 0 < \rho (c, j, \varepsilon) < 2 (1 - \varepsilon) \).

The Newton-Raphson flavor of GNNB might be useful to prove its convergence to the optimum of the surrogate risk at hand (\( \varepsilon_S \)), yet the original boosting theory is more demanding than "mere" convergence to global optimum: it requires guaranteed convergence rates under weak assumptions about each iteration.

### 4.2 GNNB boosts the surrogate risks

We consider the following Weak Learning Assumption about GNNB:

**WLA** There exist constants \( \varrho > 0, \vartheta > 0 \) such that at any iterations \( c, t \) of GNNB, index \( j \) returned by Wic is such that the following holds:

\[
\sum_{i:j \sim S, k} w_i \geq \frac{\vartheta}{\varphi (0) - \varphi (1/2)} \quad (i)
\]

\[
|\tilde{p}_w (y | j \neq y_i, j \sim S, k) - 1/2| \geq \vartheta \quad (ii)
\]

Requirement (ii) corresponds to the usual weak learning assumption of boosting [9], [14], [5]: it postulates that the current normalized weights in the inverse neighborhood of example \( j \) authorize a classification different from random by at least \( \vartheta \). GNNB uses unnormalized weights that satisfy \( (1/n_j) \sum_{i:j \sim S} w_i \in [0, 1/(\varphi (0) - \varphi (1/2))] \); requirement (i) thus implies that the unnormalized weights in the inverse neighborhood must not be too small. Intuitively, such a condition is necessary as unnormalized weights of minute order would not necessarily prevent (ii) to be met, but would impair the convergence of GNNB given the linear dependence of \( \delta_j \) in the unnormalized weights. Notice also that unnormalized weights are all the smaller as examples receive the right labels: the fact that requirement (i) becomes harder to be met simply means that GNNB approaches the optimum sought. At the beginning of GNNB, the initialization with the null leveraging vectors \((\alpha_j = 0, \forall j)\) guarantees that we can pick in (i) \( q = 1/2 \) everywhere.

The analysis we carry out is a bit more precise than usual boosting results: instead of giving, under the WLA, a lowerbound on the number of iterations needed to drive down the surrogate or empirical risks down some user-fixed threshold \( \tau \), we rather provide a lowerbound on the total number of weight updates, for each class \( c \). This number, \( \ell (T, c) \), integrates the total number of boosting iterations and the size of inverse neighborhoods used. It is important to integrate these sizes since there is obviously a big difference for convergence between leveraging an example which votes for many others in "dense" parts of the data, and leveraging one which votes for none. Our main result is split in two. The first focuses on the surrogate risk, the second on the empirical risk. Let us define:

\[
\phi_c (S) = \sum_x \tilde{p}_S (x_0) x_0 \tilde{p}_S (x = 1) \quad (19)
\]

\[
\Delta_\varphi (S, \tau, c) = \phi_c (S) - ((1 - \tau) \varphi (1/2) + \tau \varphi (0)) \quad (20)
\]

\[
\Delta \varphi (S, \tau, c) = \phi_c (S) - \varphi (1 - \tau) / 2 \quad (21)
\]

\[
\Delta_\varphi (S, \tau, c) \quad \text{and} \quad \Delta \varphi (S, \tau, c)
\]

are differences between average values of \( \varphi \) taking values within \( \pm (\varphi (0) - \varphi (1/2)) \). We now state our main result on GNNB.

**Theorem 4:** Assume the WLA holds, and let \( \tau \in [0, 1] \). Suppose we run GNNB so that, \( \forall \ell (T, c) \) meets:

\[
\ell (T, c) \geq \frac{\Delta_\varphi (S, \tau, c) (\varphi (0) - \varphi (1/2))}{8 \varepsilon (1 - \varepsilon) \vartheta / \rho (c, j, \varepsilon) \times \rho (c, j, \varepsilon)} \quad (22)
\]

Then the leveraged \( k \)-NN \( H \) learned by GNNB satisfies:

\[
\varepsilon_S (H (h_{t+1} | c)) - \varepsilon_S (h_{t+1} | c) \leq \varepsilon_S \times \rho (c, j, \varepsilon) \quad (23)
\]

**Proof:** We craft a negative upperbound for the variation of the surrogate risk at hand (2) between two successive iterations, say \( t \) and \( t+1 \). To keep references clear, we replace the index \( j \) of the example returned by Wic by \( e (t) \). We have:

\[
\varepsilon_S (h_{t+1} | c) - \varepsilon_S (h_{t+1} | c)
\]

\[
= 1 - m \sum_i \psi (y_{ic} h_{t+1} | c (x)) - m \sum_i \psi (y_{ic} h_{tc} | c (x))
\]

\[
= 1 - m \sum_i \psi (0) \psi (y_{ic} h_{t+1} | c (x)) - \sum_i \psi (y_{ic} h_{tc} | c (x))
\]

\[
= 1 - m \sum_i \psi (0) \psi (y_{ic} h_{t+1} | c (x)) - \sum_i \psi (y_{ic} h_{tc} | c (x))
\]

where \( \tilde{\psi} (x) \approx \psi (x) \hat{(x)} \) and \( \psi (x) \) comes from Lemma 1 in [14]. We have \( \tilde{\psi} (x) = - \tilde{\psi} (x) \varphi (0) - \varphi (1/2) \), implying \( \tilde{\psi} (x) = \hat{(x)} \varphi (x) \). Thus:

\[
\tilde{\psi} (x) = - \varphi (0) \varphi (1/2) \psi (x) \quad (25)
\]

Furthermore,

\[
\tilde{\psi} (y_{ic} h_{t+1} | c (x))
\]

\[
= 1 - \varphi (0) \varphi (1/2) \psi (y_{ic} h_{t+1} | c (x)) - \psi (y_{ic} h_{tc} | c (x))
\]

\[
= w_{t+1} | c (x) \quad (26)
\]
and \(\nabla_{\psi_{\phi}}^{-1}(-y_{ic}h_{tc}(x)) = w_{ti}\) as well, so that, using (25) and (26), we can simplify (24) as follows:

\[
\varepsilon_{S}^{\psi}(h_{t+1}, c) - \varepsilon_{S}^{\psi}(h_{tc}, c) = \frac{1}{m} \sum_{i : c(t) \sim s, k} \left( \sum_{i : c(t) \sim s, k} D_{\psi_{\phi}}^{e}(0)|w_{i(t+1)}| - \sum_{i : c(t) \sim s, k} D_{\psi_{\phi}}^{e}(0)|w_{it}| \right)
\]

\[
= -\frac{1}{m} \sum_{i : c(t) \sim s, k} \left\{ \tilde{\psi}_{\phi}(w_{i(t+1)}) - \tilde{\psi}_{\phi}(w_{it}) - w_{i(t+1)} \nabla_{\psi_{\phi}}^{\psi}(w_{i(t+1)}) + w_{it} \nabla_{\psi_{\phi}}^{\psi}(w_{it}) \right\}. 
\]

We lowerbound the divergence term, starting by an important property for \(\psi_{\phi}\). We show that a differentiable function \(\psi = \omega \text{ strongly smooth [19]}\) iff there exists some \(\omega > 0\) such that \(D_{\psi}(x')|x| \leq \frac{\omega}{2}(x'-x)^2, \forall x, x'.\)

**Lemma 2:** For any permissible \(\phi, \psi_{\phi}\) is \(H^{e}_{\psi_{\phi}}\) strongly smooth, where \(H^{e}_{\psi_{\phi}}\) is defined in (8).

**Proof:** Taylor-Lagrange remainder brings that there exists some \(x'' \in (x, x')\) such that

\[
D_{\psi_{\phi}}(x') = \frac{1}{2}(x-x')^2H_{\psi_{\phi}}(x'') \leq \frac{1}{2}(x-x')^2H^{e}_{\psi_{\phi}}
\]

(we used (8)). This proves Lemma 2.

From [19] that \((\psi_{\phi})^{*}\) is \((H^{e}_{\psi_{\phi}})^{-1}\) strongly convex; so,

\[
\tilde{\psi}_{\phi}(w) = -\frac{1}{2H^{e}_{\psi_{\phi}}}w^2 \text{ is convex.}
\]  

Any convex function \(\varphi\) satisfies \(\varphi(w') \geq \varphi(w) + \nabla_{\varphi}(w)(w'-w), \forall w, w'.\) We apply this inequality taking as \(\varphi\) the function in (28), \(w = w_{ti}\) and \(w' = w_{i(t+1)}\). We sum for each \(i\) such that \(c(t) \sim s, k\):

\[
\sum_{i : c(t) \sim s, k} D_{\psi_{\phi}}^{e}(w_{i(t+1)})|w_{ti}| \geq \frac{1}{2H^{e}_{\psi_{\phi}}} \sum_{i : c(t) \sim s, k} (w_{i(t+1)} - w_{it})^2. 
\]

Finally, Cauchy-Schwartz inequality yields:

\[
\sum_{i : c(t) \sim s, k} (y_{ic}y_{ic}(c))^{2} \sum_{i : c(t) \sim s, k} (w_{i(t+1)} - w_{it})^2 \geq \left( \sum_{i : c(t) \sim s, k} y_{ic}y_{ic}(c)(w_{i(t+1)} - w_{it}) \right)^2. 
\]

Fix for short \(u = \sum_{i : c(t) \sim s, k} w_{i(t+1)}y_{ic}(c)|y_{ic}(c)|.\) Plugging altogether (27), (29) and (30), we obtain the following upperbound for \(\varepsilon_{S}^{\psi}(h_{t+1}, c) - \varepsilon_{S}^{\psi}(h_{tc}, c):\)

\[
\varepsilon_{S}^{\psi}(h_{t+1}, c) - \varepsilon_{S}^{\psi}(h_{tc}, c) \leq \frac{\left( u - \sum_{i : c(t) \sim s, k} w_{i(t+1)}y_{ic}(c)e_{\phi}(c) \right)^{2}}{2H^{e}_{\psi_{\phi}} m \sum_{i : c(t) \sim s, k} (y_{ic}(c)e_{\phi}(c) - \delta_{e_{\phi}}(c))^{2}} - \frac{\delta_{e_{\phi}}(c)u}{m}.
\]

(31) \(\Delta_{i}(u)\) takes its maximum value for \(u = u^{*} = \eta(c, e(t)) - H^{e}_{\psi_{\phi}}n_{e}(c)e_{\phi}(c),\) for which we have: \(\Delta_{i}(u^{*}) = (1/2)H^{e}_{\psi_{\phi}}n_{e}(c)e_{\phi}(c)\delta_{e_{\phi}}(c) - (2\varepsilon_{c}(e(t)))/(H^{e}_{\psi_{\phi}}n_{e}(c))\).

We pick \(\delta_{e_{\phi}}(c)\) as in (10), i.e., \(\delta_{e_{\phi}}(c) = 2(1 - \varepsilon)\eta(c, e(t))\), for \(\varepsilon \in (0, 1)\). This yields:

\[
\Delta_{i}(u) \leq \Delta_{i}(u^{*}) = -2\varepsilon(1 - \varepsilon)\eta^{2}(c, e(t)) \frac{H^{e}_{\psi_{\phi}}n_{e}(c)}{H^{e}_{\psi_{\phi}}n_{e}(c)}. 
\]

(32) We now show that the WLA implies a strictly positive lowerbound on the absolute value of edge \(\eta(c, e(t))\). Letting \(I_{n}\) be the indicator function, we have \(\hat{p}_{w}y_{ic}(c) = y_{ic}(c)e_{\phi}(c) = (\sum_{i : c(t) \sim s, k} w_{i(t+1)}y_{ic}(c)e_{\phi}(c)) / (\sum_{i : c(t) \sim s, k} w_{i(t+1)}y_{ic}(c)e_{\phi}(c)),\)

and since \(I_{n}y_{ic}(c) \in [-1, 1] - y_{ic}(c)e_{\phi}(c)\) we obtain after simplification:

\[
\hat{p}_{w}y_{ic}(c) = \frac{\eta(c, e(t))}{2(\sum_{i : c(t) \sim s, k} w_{i(t+1)}y_{ic}(c)e_{\phi}(c))}. 
\]

Using statement (ii) in the WLA, this equality brings \(|\eta(c, e(t))| \geq 2\delta \sum_{i : c(t) \sim s, k} w_{i(t+1)}y_{ic}(c)e_{\phi}(c)).\) Using statement (ii) in the WLA, we finally obtain:

\[
|\eta(c, e(t))| \geq 2\delta \frac{\delta_{e_{\phi}}(c)}{H^{e}_{\psi_{\phi}}}(\phi(0) - \phi(1/2)). 
\]

(33) Plugging (33) into (32), and the resulting inequality into (31), we obtain:

\[
\varepsilon_{S}^{\psi}(h_{t+1}, c) - \varepsilon_{S}^{\psi}(h_{tc}, c) \leq -8\varepsilon(1 - \varepsilon) \frac{n_{e}(c)\phi^{2}d^{2}}{mH^{e}_{\psi_{\phi}}(\phi(0) - \phi(1/2))^{2}}. 
\]

(34) At the initialization, all leveraging coefficients \(\alpha_{j}\) equal the null vector, and so the corresponding surrogate risk equals \(\psi(0).\) To guarantee that \(\varepsilon_{S}^{\psi}(h_{tc}, c) \leq (\varepsilon_{S}^{\psi})^{*} + \tau\) under the WLA, for some \(\tau \in [0, \psi(0)],\) it is thus sufficient to have:

\[
\sum_{t \sim 1} \frac{1}{m} \frac{\left( \psi(0) - \left( \varepsilon_{S}^{\psi})^{*} - \tau\right)H^{e}_{\psi_{\phi}}(\phi(0) - \phi(1/2))^{2} \right)}{8\varepsilon(1 - \varepsilon)d^{2}} \geq \frac{\tau}{m}. 
\]

This inequality leads to the statement of the Theorem, provided we remark the three following facts. The first one is proven in the following Lemma.

**Lemma 3:** We have \((\varepsilon_{S}^{\psi})^{*} = (\phi(0) - \phi(\epsilon(S)))/\phi(0) - \phi(1/2)),\) where \(\phi_{c}(S)\) is defined in (19).

**Proof:** From Lemma 1, we have \(\nabla^{-1}(x) = -\nabla^{-1}(-x),\) with which we obtain after few derivations:

\[
\arg \min_{S_{\phi}} \varepsilon_{S}^{\psi}(h, c) = \nabla \hat{y}_{ic}(y_{ic} = 1|x), S_{\phi} \text{ is }
\]

0162-8828 (c) 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
the subset of \( S \) whose observations match \( x \). Then, we compute \( \varepsilon^\phi_{S}(h,c) \) with this value for \( h \), which, after simplification using Legendre conjugates, brings \( E_{S[x]}[\psi(y_{ic,h}(x))] = (\phi(0) - \phi(\hat{p}_{S[x]}(y_{ic} = 1|x))/\phi(0) - \phi(1/2)) \). Finally, we average this over all distinct observations in \( S \) to obtain Lemma 3.

The last two facts that lead to the statement of the Theorem are simpler: we indeed have \( \sum_{i=1}^{T} n_{e(t)} = \ell(T,c) \), and \( \psi_{0}(0) = (\phi(0) - \phi(1/2))/\phi(0) - \phi(1/2) = 1 \). This concludes the proof of Theorem 4.

4.3 GNNB boosts the empirical risk

The following bound holds on the empirical risk.

**Corollary 1:** Assume the WLA holds, and let \( \tau \in [0,1] \). Suppose we run GNNB so that, \( \forall e, \ell(T,c) \) means:

\[
\ell(T,c) \geq \frac{\Delta_{h}(S,\tau,c)(\phi(0) - \phi(1/2))}{8c(1-\varepsilon)\varrho^{2}\varrho^{2}} \times m
\]

Then the leveraged \( k \)-NN \( H \) learned by GNNB satisfies:

\[
\varepsilon_{H}^{\psi,h}(H) \leq \varepsilon_{H}^{\psi,h} + \tau \tag{36}
\]

**Proof:** Following [7], let us define \( H(\varepsilon) = \inf_{\delta \in \mathbb{R}} \{\phi(\delta) + (1-\varepsilon)\phi(\delta - \delta)\} \), \( \psi_{\text{in}}(\varepsilon) = \psi(0) - H((1+\varepsilon)/2) \), with \( \varepsilon \in [0,1] \) and \( \varepsilon \in [-1,1] \). We have:

\[
H(\varepsilon) = \inf_{\delta \in \mathbb{R}} \left\{ \phi(\delta + (1-\varepsilon)\phi(\delta + (1-\varepsilon)\phi(\delta) + \phi(0) \right\}
\]

\[
= \inf_{\delta \in \mathbb{R}} \left\{ \phi(\delta - \delta) + (1-\varepsilon)\phi(\delta) + \phi(0) \right\}
\]

\[
= \phi(0) - \phi(1/2)
\]

\[
= -\phi(0) + \phi(0) \tag{37}
\]

Here, (37) follows from Lemma 1, and (38) follows from the fact that \( \phi \) is convex and lower semicontinuous. We thus have:

\[
\psi_{\text{in}}(\varepsilon) = \phi((1+\varepsilon)/2) - \phi(1/2)
\]

It is proven in [7], Theorem 1, that:

\[
\psi_{\text{in}}(\varepsilon) = \left( \varepsilon_{S}^{\psi,h}(h,c) - (\varepsilon_{S}^{\psi,h})_{c} \right)
\]

The argument of \( \psi_{\text{in}} \) is in \([0,1]\). On this interval, \( \psi_{\text{in}} \) admits an inverse because \( \phi \) admits an inverse on \([0,1/2]\). To ensure \( \varepsilon_{S}^{\psi,h}(h,c) \leq (\varepsilon_{S}^{\psi,h})_{c} + \tau \), it is thus equivalent to ensure \( \varepsilon_{S}^{\psi_{0,h}}(h,c) - (\varepsilon_{S}^{\psi_{0,h}})_{c} \leq \psi_{\text{in}}(\tau) \). There remains to combine (39) and (22) to obtain the statement of Corollary 1.

4.4 GNNB is universally consistent

We analyze GNNB in the setting where WIC yields the leveraging of a subset of \( m' < m \) examples out of the \( m \) available in \( S \). This setting is interesting because it covers the optimization of GNNB in which we repeatedly leverage the most promising example, for example from the standpoint of \( |\delta_{j}| \). We call GNNB* this variation of GNNB. We assume that \( S \) is sampled i.i.d. according to some fixed density. The following (weak) universal consistency result on GNNB is not surprising, as NN approaches were the first to be proven consistent [20], and there have been since a wealth of weak and strong related universal consistency results [17]. The result also applies to UNN [4].

**Lemma 4:** Provided \( T,k \to \infty \), \( k/m' \to 0 \) and \( m'/m \to 0 \), GNNB* is (weak) universally consistent: its expected Hamming risk converges to the Hamming risk of Bayes rule.

**Proofsketch:** The proof gathers several blocks, the first of which is the fact that the empirical minimization of surrogate BCL \( \psi_{l} \) in an NN approach amounts to a maximum likelihood fitting of class posteriors [14] (Lemma 4). Indeed, after dropping temporarily the class index \( c \) to focus first on a single class, the corresponding empirical risk \( \varepsilon_{S}^{\psi_{l}}(h) = \sum_{h} \hat{p}([x,y]) / \psi_{l}(y/h(x)) \) (13) satisfies (see Theorem 2):

\[
\varepsilon_{S}^{\psi_{l}}(h) \propto \sum_{i=1}^{n} \hat{p}_{i}([x,y]) \frac{\hat{p}([x,y])}{\hat{p}_{i}[y]} D_{p}(p(y)||\hat{p}) \tag{40}
\]

Here, \( v \) is the number of Voronoi cells \( V_{1} \), \( l = 1,2,...,v \); \( \hat{p}_{i}([x,y]) = \sum_{V_{j} \in V_{1}} \hat{p}([x,y])/\hat{p}_{j} \) and \( \hat{p}_{i} = \hat{p}_{j} \) for cell \( V_{j} \). Second, the right-population minimizer of any Bregman divergence is always the arithmetic average [21] (Proposition 1): at the minimum of each \( \varepsilon_{l} \) in (40), \( \hat{p}_{i} = \hat{p}_{j} = \hat{p}_{V_{j}} \). Lastly, Corollary 6.2 in [17] makes that a sufficient condition for the (weak) universal consistency of GNNB with respect to class \( c \), and by extension to all classes for Hamming risk.

5 DISCUSSION

We chose not to normalize permissible functions, i.e. typically ensuring \( \phi(1/2) = 1 \) and \( \phi(0) = 0 \), because normalization would reduce the number of BCL that can be generated. For example, out of the two in rows B and C in Table 1, the classical form of the logistic loss in B would disappear. Bounds in (22) and (35) advocate for a simple implementation of WIC: since the number of examples leveraged equals, on average, \( \ell(T,c)/k \), we should put emphasis on leveraging examples with large inverse neighborhoods.

Our results call for several technical comparisons between GNNB, UNN and mathematical greedy algorithms [1]. Let us define:

\[
\pi^{*} = (\phi(0) - \phi(1/2))^{2} H^{*}_{\phi} / 2 \tag{41}
\]
and let us respectively define \( \pi(\varepsilon) \) and \( \pi'(\varepsilon) \) the terms factoring \( m(\vartheta^2 g^2)^{-1} \) in (22) and (35). Because 
\[
\Delta_\varphi(S, \tau, c) \leq \Delta_\varphi(S, \tau, \alpha) \leq \varphi(0) - \varphi(1/2),
\]
it comes \( \pi(1/2) \leq \pi'(1/2) \leq \pi^*. \) Table 1 provides examples of values for \( \pi^* \) for different choices of \( \varphi; \) they are small, in [1/8, 1/16]. Hence, when \( \varepsilon = 1/2, \) a sufficient number of weight updates to ensure (23) and (36) is 
\[
\ell^*(T, c) = (m/8) \times (\vartheta^2 g^2)^{-1}.
\]
This happens to be a very reasonable constraint, given that the range of \( \delta_j \) is of minute order compared to that in UNN, where \( \delta_j \) can take on infinite values.

There is more: let \( \ell_{\text{GNNB}}(T, c) \) and \( \ell_{\text{UNN}}(T, c) \) denote the number of weight updates ensuring (36) (and thus ensuring (23) as well), respectively 
for GNNB and UNN. Inspecting Theorem 2.3 in [4] reveals that we have 
\[
\ell_{\text{GNNB}}(T, c) = \Theta(\ell_{\text{UNN}}(T, c)/(\varepsilon(1-\varepsilon))).
\]
Hence, convergence rates of GNNB compete with those known for UNN.

Mathematical greedy algorithms [1] have a very wide scope, and they can be specialized to statistical learning with a high-level scheme which is close to the iterative scheme of boosting algorithms. Situating GNNB with respect to them is thus interesting and reveals quite a different picture, from the computational and convergence rate standpoints. These greedy algorithms are indeed computationally expensive, requiring at each iteration a local optimization of the classifier that GNNB does not require. Regarding convergence rates, the bound most relevant to our setting can be stated as follows, omitting unnecessary technical details and assumptions [1] (Theorem 3.1 and its proof): after \( t \) iterations, the squared risk of the greedy output is no more than 
\[
\tau(t) = \beta ((k/n^2 t) + (t \ln(m)/m)),
\]
for some \( \kappa, \beta \) that meet in general \( \kappa \gg m \) and \( \beta > 10^4. \) This bound takes its minimum for some \( t^* \) which is \( \gg m \) in general. Even for this large \( t^* \), the corresponding upperbound on the squared risk, 
\[
\tau(t^*) = 2\beta\sqrt{\kappa \ln(m)/m},
\]
is significantly weaker than the guarantees of Theorem 4 and Corollary 1. Obviously however, our bounds rely on the WLA.

6 EXPERIMENTS

6.1 Domains and metrics

Experiments have been performed on a dozen domains summarized in Table 2. We have split the domains in small and large domains. Large domains have a significantly larger number of examples and classes. We refer the reader to the UCI machine learning repository for the related domains. We give a brief description of the “large” domains. The Caltech [12] domain is a collection of 30607 images of 256 object classes. We adopt the Fisher vectors [22] encoding in order to describe these images as features vector. Fisher Vector are computed over densely extracted SIFT descriptors and local color features, both projected with PCA in a sub space of dimension 64. Fisher Vectors are extracted using a vocabulary of 16 Gaussian and normalized separately for both channels and then combined by concatenating the two features vectors. This yields a 4K dimensional features vector. The SUN [4], [13] domain is a collection of 108656 images divided into 397 scenes categories. The number of images varies across categories, but there are at least 100 images per category. Each observation is represented as feature vector computed in the same way as for Caltech. Experiments are performed on a classical five-fold cross-validation basis, except for the large domains Caltech and SUN for which we have adopted the standardized approaches to use 30 (for Caltech) and 50 (for SUN) random images from each class to train classifiers and the remaining for testing.

We consider three types of metrics: the accuracy, which is one minus the Hamming risk (12, 13) and which is directly optimized by GNNB (Corollary 1), the recall and the F-measure.

6.2 Algorithms

To make an extensive analysis of the performances of GNNB, we have evaluated on small domains twenty-two (22) algorithms, on each of the three metrics. The version of GNNB used is GNNB(log) (Row B in Table 1) with values of \( k = 5, 10, 20, 50. \) Contenders of GNNB can be put in five categories: ordinary nearest neighbors, universal nearest neighbors (UNN), stochastic gradient descent algorithms, (Ada)boosting algorithms and support vector machines.

Ordinary nearest neighbors, NN, and UNN(log) were tested with \( k = 5, 10, 20, 50. \) UNN performs for this choice of BCL approximations to the optimal boosted updates [4]. We used the simplest, non optimized WiC in UNN and GNNB, which returns index \( t \mod m. \)

We considered Stochastic Gradient Descent (SGD) [22], [23], [24], with four varying number of iterations. In the first, referred to as SGD, the number of iterations is equal to that of GNNB and UNN. In the second, SGD2, number of iterations for SGD is fixed to be the “equivalent” to that of UNN and GNNB. Indeed, each iteration of SGD contributes to classify all examples in the training sample, while each iteration of UNN or

<table>
<thead>
<tr>
<th>category</th>
<th>name</th>
<th>m</th>
<th>C</th>
<th>ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>small</td>
<td>Liver</td>
<td>345</td>
<td>2</td>
<td>[11]</td>
</tr>
<tr>
<td></td>
<td>Ionosphere</td>
<td>351</td>
<td>2</td>
<td>[11]</td>
</tr>
<tr>
<td></td>
<td>Pima</td>
<td>768</td>
<td>2</td>
<td>[11]</td>
</tr>
<tr>
<td></td>
<td>Scene</td>
<td>2407</td>
<td>6</td>
<td>[11]</td>
</tr>
<tr>
<td></td>
<td>Satellite</td>
<td>6435</td>
<td>6</td>
<td>[11]</td>
</tr>
<tr>
<td></td>
<td>Segment</td>
<td>2310</td>
<td>7</td>
<td>[11]</td>
</tr>
<tr>
<td></td>
<td>Cardio</td>
<td>2126</td>
<td>10</td>
<td>[11]</td>
</tr>
<tr>
<td></td>
<td>OptDig</td>
<td>5620</td>
<td>10</td>
<td>[11]</td>
</tr>
<tr>
<td>large</td>
<td>Caltech</td>
<td>30607</td>
<td>256</td>
<td>[12]</td>
</tr>
<tr>
<td></td>
<td>SUN</td>
<td>108656</td>
<td>397</td>
<td>[13]</td>
</tr>
</tbody>
</table>

TABLE 2

Domains used in our experiments, ordered in increasing number of classes, and then examples.
GNNB contributes to classify $\theta(k)$ examples only. Thus, we need $\theta(m/k)$ iterations on UNN or GNNB for the classification of all examples to be eventually impacted. So, if $T$ is the total number of boosting iterations in UNN and GNNB, then we perform $T \times k/m$ iterations of SGD. The two last runs of SGD, hereafter noted SGD$_3$ and SGD$_4$, consider a larger number of iterations, twice the size of the training set in SGD$_3$ and three times in SGD$_4$. With those runs, we wanted to capture “limit” performances of SGD.

We also considered ADABOOST [5], with four different flavors. In ADABOOST$_{t2}$ (resp. ADABOOST$_{t3}$), the weak learner is C4.5 [25] with depth-2 (resp. depth-3) trees. C4.5 is a powerful weak learner: it repeatedly minimizes the expected $-\phi$ in row B of Table 1. Again, the weak learner (VIC) used in GNNB and UNN is deliberately not optimized at all. For this reason, we have also tested ADABOOST with a non-optimized weak learner, which returns random trees. In ADABOOST$_{t3}$, these trees have depth 3, and in ADABOOST$_{t4}$, these trees have unbounded depth. In all four flavors of ADABOOST, the number of boosting rounds equals that of GNNB and UNN.

We have also considered two flavors of support vector machines, the first of which is affordable on small domains (but out of reach on our largest domains), non-linear SVM with radial basis function kernel in which the regularization parameter and the bandwidth are further optimized by a five-fold cross-validation on the training sample. We refer to them as SVM$_{RBF}$. The second flavor is linear SVM, SVM$_{L}$.

On large domains, we have tested GNNB against the contenders that scored top in the small domains or were easily scalable to large domains: NN, UNN, SGD. We have also tried SVM$_{LLC}$, that is, linear SVM with locality-constrained linear coding LLC [26].

6.3 Results on small domains

6.3.1 Results on average metrics

Figure 3 presents the average results obtained for the 22 algorithms on the 3 metrics. Over all metrics, one can notice that the algorithms cluster in 3 groups.

The first is the group of the best performing algorithms, with non-linear and mostly optimized large margin algorithms: SVM$_{RBF}$, GNNB (all $k$s), UNN (all $k$s), ADABOOST+C4.5, and NN with $k=5,10,20$. The second group performs not as well as the first, with mostly linear classification algorithms: SVM$_{L}$, all SGD algorithms and NN with $k=50$. The last group perform the worst of all, containing randomized large margin classification: ADABOOST with random trees.

Several observations can be made. First, the performances of all nearest neighbor methods (GNNB, UNN, NN) decrease with $k$, in the range of values selected. Second, boosting nearest neighbors (GNNB, UNN) dampens the degradation of performances. Third, GNNB is the best of all kinds of nearest neighbor methods, from the standpoint of all metrics.

In fact, GNNB performs on par with SVM$_{RBF}$, for a wide range of $k$ (5, 10, 20). The comparison with ADABOOST$_{t3}$ and ADABOOST$_{t4}$ is clear and final, as regardless of $k$ and for all metric, GNNB is better by more than 0.2 points on average; finally, GNNB performs also slightly better than ADABOOST+C4.5 (for $k=5,10,20$). These are good news, first because GNNB is not optimized as ADABOOST+C4.5 is (for example from the standpoint of the weak learner), and second because GNNB is the lightest machinery among all, and so the easiest to scale to large domains.

6.3.2 Ranking results

To drill down into these general results, we have also computed the global ranking results of each algorithm, recording the number of times each ranked first, second, third and so on, on the 9 domains. These results (Figure 4), yield the following observations.

First, there is a subgroup in the group of the best performing algorithms according to the average metrics, which is the best according to ranking: SVM$_{RBF}$ and GNNB ($k=5,10,20$). In this group, it appears that GNNB tends to be ranked higher than SVM$_{RBF}$, for a wide range of $k$ (5, 10, 20), and this is particularly visible for F-measure and recall. From the recall standpoint, GNNB is almost always in top-tier results, while SVM$_{RBF}$ is more often in the second-tier.
Second, SGD performs poorly from the ranking standpoint, as all flavors mostly score among the third-tier results. We also observe that SGD performances are not monotonic with the number of iterations, as SGD\(_1\) performs the best of all, both from the average and ranking standpoints. Linear classification methods tend to perform poorly, as displayed in the average and ranking standpoints. Linear classification algorithms, ranging from linear to non-linear, perform on par with or better than optimized large margin non-linear algorithms (SVM\(_{RBF}\), ADABOOST+C4.5).

6.3.3 Classification patterns

The algorithms we have tested on small domains are representative of major families of supervised classification algorithms, ranging from linear to non-linear, induced to non-induced, including large margin classification methods, stochastic algorithms, and so on. To get a qualitative picture of the performances of GNNB, we have learned a manifold on the algorithms’ results, one for each of the three metrics, as follows.

To get rid of the quantitative differences, we have normalized results to zero mean and unit standard deviation in each domain. Then, a manifold was learned using a standard procedure, with the normalized cosine similarity measure, and computing the second and third leading eigenvector of the Markov chain from the associated similarity matrix [28].

The corresponding manifolds are displayed in Figure 6, using a focus+context display [27] in which
and the rest of the algorithms (center). The accuracy and F-measure plot make a clear distinction between non-linear large margin “optimized” (down-right), non-linear large margin “random” (down-left) and linear (up). Looking at nearest neighbor algorithms as \( k \) increases reveals that boosted nearest neighbor algorithms (UNN, GNNB) tend to behave more and more like large margin classification algorithms as \( k \) increases, while vanilla NN tends to behave more and more like linear classification algorithms as \( k \) increases. This observation for NN is consistent with the simple example that sampling two spherical Gaussians with identical variance (one for each class) makes a non-linear frontier for \( k, m \ll +\infty \), which tends to a linear one as both parameters tend to +\( \infty \).

### 6.3.4 Training times

We have computed the training times for GNNB (all \( k \)s), SVM\(_{RBF} \) and ADABOOST+C4.5 (depth-3 trees), that belong to the top-5 or top-6 algorithms in terms of average metric performances. We have computed the ratio between training times for each domain and each value of \( k \), for SVM\(_{RBF} \) to GNNB, and ADABOOST+C4.5 to GNNB. As already displayed for UNN [4], the ratios are clearly in favor of GNNB. We obtained a synthetic and accurate picture of these advantages by regressing the ratio against \( 1/k \), that is, computing the regression coefficients \( a, b \) for \( \rho = (a/k) + b \). Here, \( \rho \) is e.g. the ratio for the SVM\(_{RBF} \) training time to GNNB training time, averaged over all domains, and then computed for each \( k \). The results, that we give with the coefficient of determination \( r^2 \), are (t.t. = training time):

\[
\frac{\text{t.t.}(\text{SVM}_{RBF})}{\text{t.t.}(\text{GNNB})} \approx \frac{851}{k} + 49 \quad (r^2 = 0.96),
\]

\[
\frac{\text{t.t.}(\text{ADABOOST+C4.5})}{\text{t.t.}(\text{GNNB})} \approx \frac{9547}{k} + 398 \quad (r^2 = 0.97).
\]

These regressions mean that, regardless of the value of \( k \), SVM\(_{RBF} \)'s training time is at least roughly 50 times that of GNNB, while ADABOOST+C4.5's training time is at least roughly 400 times that of GNNB. These ratios are in good agreement with those observed in favor of UNN against SVM\(_{RBF} \) and ADABOOST+stumps [4].

---

**TABLE 3**

Performance of our divide-and-conquer approach on large domains for GNNB(log), using top-1 and top-5 accuracies.

<table>
<thead>
<tr>
<th>( \text{top-1 accuracy} \times 100 ), Caltech</th>
<th>( \text{top-1 accuracy} \times 100 ), SUN</th>
<th>( \text{top-5 accuracy} \times 100 ), SUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{iteration} \ t )</td>
<td>( n )</td>
<td>( n )</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>23.63</td>
<td>26.40</td>
</tr>
<tr>
<td>10</td>
<td>23.85</td>
<td>26.63</td>
</tr>
<tr>
<td>100</td>
<td>25.64</td>
<td>27.98</td>
</tr>
<tr>
<td>1000</td>
<td>29.38</td>
<td>31.52</td>
</tr>
</tbody>
</table>

---

Fig. 6. Manifold classification patterns for the accuracy (up, commented), F-measure (bottom left) and Recall (bottom right), for all 22 algorithms (see text); colors in hexagons cluster types of algorithms: red = GNNB, yellow = SVM, pink = ADABOOST, cyan = SGD, blue = UNN, green = NN (see text and [27] for details).
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2014.2307877, IEEE Transactions on Pattern Analysis and Machine Intelligence

<table>
<thead>
<tr>
<th></th>
<th>NN</th>
<th>GNNB</th>
<th>SGD1</th>
<th>SVMLLC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>f</td>
<td>4K</td>
<td>4K</td>
<td>4K+4K</td>
</tr>
<tr>
<td>Acc.</td>
<td>25.50</td>
<td>36.40</td>
<td>36.00</td>
<td>27.99</td>
</tr>
<tr>
<td>F-m.</td>
<td>20.97</td>
<td>29.24</td>
<td>30.87</td>
<td>24.00</td>
</tr>
<tr>
<td>Rec.</td>
<td>17.13</td>
<td>31.47</td>
<td>31.35</td>
<td>22.00</td>
</tr>
</tbody>
</table>

**TABLE 4**

Results on Caltech (accuracy, F-measure and recall are \( \times 100 \)). \( f \) is the number of features, and \( k = 200 \) for NN, GNNB.

### 6.3.5 Summary for small domains

The results obtained on small domains bring the following general observations. First, GNNB scores among the top algorithms and performs on par with, or better than, optimized machineries like non-linear SVM or ADABOOST+trees, and it beats these latter approaches, from the training times standpoint, by factors that range from tens to thousands of times. These good performances go hand in hand with the desirable property that results are stable against reasonable variations of \( k \), which is not the case for UNN.

### 6.4 Results on large domains

We have used the instantiation of SGD that performed the best on small domains, SGD1, and the number of iterations of GNNB and SGD1 is 6000. We split the analysis between the comparison of GNNB vs UNN, and GNNB vs the rest of the algorithms.

#### 6.4.1 A divide-and-conquer optimization of GNNB

It is well known that NN classifiers suffer of the curse of dimensionality [17], so that the accuracy can decrease when increasing the size of descriptors. This may also affect GNNB, in particular on large domains like SUN and Caltech. Fisher vectors employ powerful descriptors but they generate a space with about 4K dimension for 32 gaussians, which could impair GNNB performance. Our approach relies on a property of classification-calibrated losses that one can get simple posterior estimators from the classifier’s output, based on the matching posterior \( \hat{\phi}_{\phi,k} \) in (5) (see [8], [4], and the right plot Figure 1). The method we propose consists in (i) splitting the set of descriptors, (ii) compute posteriors over each of these sets, and finally (iii) average the posteriors over all splits. The set of Fisher descriptors is split in a regular set of \( n \in \{8, 16, 32\} \) sub-descriptors; each set is normalized in \( L_1 \) or \( L_2 \) norm. Finally, posteriors are combined linearly, with an arithmetic average.

Table 3 presents the results obtained on our large domains. Results in Table 3 show that increasing \( n \), the number of splits, always improves the performances of GNNB, in a range between 1% and 6%, the largest improvements being obtained for the largest domain (SUN). We have also checked that increasing the number of iterations still keeps this pattern, which is thus robust to both variations in \( n \) and the total number of boosting iterations \( t \). We have witnessed in some cases differences that become much more important with the increase in \( t \). For example, after 7650 iterations on Caltech, GNNB’s top-1 accuracy becomes respectively 31.91%, 33.79% or 36.13% for \( n = 8, 16 \) and \( n = 32 \).

In the following results, GNNB is ran with \( n = 32 \) splits. To remain fair with UNN, we have also carried out the same \( n = 32 \) splitting strategy, and checked that it improves the performances of UNN as well.

#### 6.4.2 Results on Caltech

The two left plots of Figure 7 display the results of GNNB vs UNN on Caltech. We have chosen to put emphasis on the relative variations of GNNB wrt UNN, to get a clean quantitative picture of the improvements. Those plots display that GNNB outperforms UNN, and this phenomenon is dampened as \( k \) increases. For \( k = 100 \), the improvement of GNNB on accuracy and recall exceed +20%, and it is reduced to +10% for \( k = 200 \). Table 4 compares GNNB to NN, SGD1 and LLC encoding for linear SVM using the same codebook as [26]. LLC produces a very large number of descriptors compared to the 4K Fisher vectors used in the other approaches, and a significant part of the improvement due to encoding comes in fact from this very large description space [29]. In order to make fair comparisons with the other techniques that rely on 4K descriptors, we have extracted the two first layers of descriptors of LLC, of size 4K and \( 4 \times 4K \), to analyze SVMLLC over 4K descriptors, \( 4 \times 4K \) descriptors and \( 4K+4 \times 4K = 5 \times 4K \) descriptors.

The accuracy results show that GNNB tops NN and SGD1, and beats SVMLLC until 16K descriptors. It is only when SVMLLC uses five times the number of descriptors of GNNB that it beats GNNB. In fact, when using the same description size as the other algorithms, LLC encoding is beaten from the standpoint of all metrics by GNNB and SGD1. SGD1 performs well from the standpoint of the F-measure, and performs on par with GNNB from the recall standpoint.

#### 6.4.3 Results on SUN

The comparison between GNNB and UNN (Figure 7, right plots) displays the same patterns as for Caltech: as \( k \) increases, the improvements of GNNB wrt UNN are dampened, yet they are now always in favor of GNNB, and the improvements are more significant.

Table 5 compares the performances of GNNB, NN and SGD1. This time, SGD1 beats GNNB from the standpoint of all metrics. This observation has to be taken with a pinch of salt, as the experimental setting for large domains disfavors GNNB. Indeed, GNNB, like UNN and NN, is a local classifier, and for such kinds of methods, the experimental setting amounts to producing random edited nearest neighbors [17] by filtering out most (≈80%) of the dataset, with consequences that are likely to be harmful as (i) drastic random

---

0162-8828 (c) 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
that the number of examples actually used by $GNNB^*$ (i.e. leveraged or reweighted) was comparable to that used by $SGD_1$, so that both algorithms had at least approximately the same amount of information for learning. This is shown in Figure 8 (left plot): we have recorded for each class the percentage of examples actually used in training by $GNNB^*$, and plotted the corresponding estimated density. The expectation of this density is roughly 40%. Thus, 40% of the 50% of each class was used in average by $GNNB^*$, i.e. $\approx 54$ examples, to be compared to the 50 used by $SGD_1$.

The right plot in Figure 8 summarizes the improvements of $GNNB^*$ with respect to $SGD_1$. One sees this time that even when the recall of $GNNB^*$ is smaller than that of $SGD_1$, the accuracy is now comparatively significantly higher. While optimizing WIC in GNNB was not the purpose of this paper, this simple experiment displays that (i) there is significant room for further improvement of GNNB while staying in the boosting/consistency regimes, and (ii) these improvements are affordable in a large scale learning setting.

7 CONCLUSION

We proposed a simple Newton-Raphson leveraging scheme for nearest neighbors to optimize any even, twice differentiable proper scoring rule, with guaranteed convergence rates under the boosting framework that compete with those known for non-gentle approaches [4]. To the best of our knowledge, these convergence rates in the boosting framework are new for gentle boosting approaches. Experiments display that GNNB significantly outperforms UNN, converging faster to better solutions. On small domains, GNNB performs on par with or better than powerful non-linear large margin learners like non-linear SVM and Adaboost+C4.5. Large domains, on which these latter approaches are ruled out for computational costs, display that GNNB provides a lightweight competitive alternative to stochastic gradient descent. A byproduct of our experiments shows that manifold learning may be useful to assess global qualitative comparisons of algorithms. As learning algorithms are rapidly becoming more numerous and complex, this may be interesting for large-scale benchmarking, and might help in the design of new algorithms.
8 ACKNOWLEDGMENTS

The authors would like to thank the reviewers for insightful comments. Codes used (GNB and manifold learning) available upon request to R. N. and M. B.

REFERENCES


Richard Nock received a PhD in Computer Science (1999) and an accreditation to lead research in computer science (HDR, 2002) from the University of Montpellier, France. His research interests include machine learning, data mining, image processing, information geometry. He won the best paper award at ECAI 2006, and his project “Computational Information Geometry” won ANR’s Digital technology Grand Award in 2013.

Wafa Bel Haj Ali defended a PhD in Computer Science in 2013 at the University of Nice-Sophia Antipolis. She previously obtained MSocs in Computer Science and Computer Engineering from PolytechNiceSophia (2009) and Ecole Nationale des Sciences de l’Informatique (Tunisia, 2008). Her research interests include biomedical image processing, machine learning and information retrieval.

Roberto D’Ambrosio received a MSc in Biomedical Engineering in 2010 from University Campus Bio-Medico di Roma. Since 2011 he has been a PhD student in Biomedical Engineering at University Campus Bio-Medico di Roma and jointly in Information and Communication Technologies at the University of Nice-Sophia Antipolis. His primary research interest include imbalanced datasets classification and machine learning.

Frank Nielsen is Full Professor of Computer Science at Ecole Polytechnique (France) and Senior Researcher at Sony Computer Science Labs, Inc. (Japan). He received his PhD in Computer Science in 1996 and an accreditation to lead research in computer science (2006) from University of Nice-Sophia Antipolis. His research interests include geometry, vision, learning, graphics, optimization. He won the best paper award at ECAI 2006.

Michel Barlaud is Full Professor of Computer Science at the University of Nice-Sophia Antipolis. A Senior Member of the Institut Universitaire de France and Fellow of the IEEE, he has co-founded and served as Associate Editor of IEEE Trans. on Image Processing. He has served in more than 200 PhD committees worldwide, and was a member of the Technical Committee of the IEEE Signal Processing Society.