Boosting Nearest Neighbors for the Efficient Estimation of Posteriors

Roberto d'Ambrosio, Richard Nock, Wafa Bel Haj Ali, Frank Nielsen, Michel Barlaud

Boosting Nearest Neighbors for the Efficient Estimation of Posteriors

Roberto D'Ambrosio[1,3], Richard Nock[2], Wafa Bel Haj Ali[3], Frank Nielsen[4], and Michel Barlaud[3,5]

[1] University Campus Bio-Medico of Rome, Rome, Italy
r.dambrosio@unicampus.it
[2] CEREGMIA - Université Antilles-Guyane, Martinique, France
rnock@martinique.univ-ag.fr
[3] CNRS - U. Nice, France
{belhajal,barlaud}@i3s.unice.fr
[4] Sony Computer Science Laboratories, Inc., Tokyo, Japan
Frank.Nielsen@acm.org
[5] Institut Universitaire de France

Abstract. It is an admitted fact that mainstream boosting algorithms like AdaBoost do not perform well to estimate class conditional probabilities. In this paper, we analyze, in the light of this problem, a recent algorithm, UNN, which leverages nearest neighbors while minimizing a convex loss. Our contribution is threefold. First, we show that there exists a subclass of surrogate losses, elsewhere called balanced, whose minimization brings simple and statistically efficient estimators for Bayes posteriors. Second, we show *explicit* convergence rates towards these estimators for UNN, for any such surrogate loss, under a Weak Learning Assumption which parallels that of classical boosting results. Third and last, we provide experiments and comparisons on synthetic and real datasets, including the challenging SUN computer vision database. Results clearly display that boosting nearest neighbors may provide highly accurate estimators, sometimes more than a hundred times more accurate than those of other contenders like support vector machines.

1 Introduction

Boosting refers to the iterative combination of classifiers which produces a classifier with reduced true risk (with high probability), while the base classifiers may be weakly accurate [**?**]. The final, *strong* classifier h, satisfies $\mathrm{im}(h) \subseteq \mathbb{R}$. Such an output carries out two levels of information. The simplest one is the sign of the output. This discrete value is sufficient to classify an unknown observation $\boldsymbol{x}$: $h(\boldsymbol{x})$ predicts that $\boldsymbol{x}$ belongs to a class of interest iff it is positive. The most popular boosting results typically rely on this sole information [**?**,**?**,**?**] (and many others). The second level is the real value itself, which carries out as additional information a magnitude which can be interpreted as a "confidence" in the classification. This continuous information may be fit into a link function

$f : \mathbb{R} \rightarrow [0,1]$ to estimate conditional class probabilities, thus lifting the scope of boosting to that of Bayes decision rule [?]:

$$\hat{\mathbf{Pr}}[y = 1|\boldsymbol{x}] = f(h(\boldsymbol{x})) \ . \quad (1)$$

To date, estimating posteriors with boosting has not met the same success as predicting (discrete) labels. It is widely believed that boosting and conditional class probability estimation are, up to a large extent, in conflict with each other, as boosting iteratively improves classification at the price of progressively overfitting posteriors [?,?]. Experimentally, limiting overfitting is usually obtained by tuning the algorithms towards early stopping [?].

Very recently, a new algorithm was proposed to leverage the famed nearest neighbor (NN) rules [?]. This algorithm, UNN, fits real-valued coefficients for examples in order to minimize a surrogate risk [?,?]. These leveraging coefficients are used to balance the votes in the final k-NN rule. It is proven that, as the number of iterations $T \rightarrow \infty$, UNN achieves the global optimum of the surrogate risk at hand for a wide class of surrogates called strictly convex surrogates [?,?]. An explicit convergence rate is obtained for the specific case of the exponential loss, under a so-called "weak index assumption" [?].

Our contribution is threefold. First, we show that there exists a subclass of surrogate losses, elsewhere called *balanced*, whose minimization brings simple and efficient estimators for Bayes posteriors (1). Second, we show explicit convergence rates for UNN for *any* such surrogate loss under a Weak Learning Assumption which parallels that of classical boosting results [?]. Third and last, we provide experiments on simulated and real domains, displaying that boosting nearest neighbors brings very good results from the conditional class probabilities estimation standpoint, *without* the overfitting problem of classical boosting approaches. A serious challenger to the popular logistic estimator for posteriors estimation also emerges, beating it by orders of magnitude on simulated data. We end up with the conclusion that learning posteriors with boosting nearest neighbors benefits from two advantages. First, the weak classifiers being simple examples, they naturally limit the risk of overfitting compared to more complex weak learners. Second, we end up learning posteriors using a *natural, fixed* topology of data, and not an *ad hoc* topology relying on an induced classifier.

The remaining of the paper is structured as follows: the next Section presents definitions, followed by a Section on convex losses and the estimation of posteriors. Then, a Section presents algorithms and results on boosting nearest neighbors. The two last Sections present experiments with discussions, and conclude.

2 Definitions

2.1 Estimation

Our setting is that of multiclass multilabel classification (See *e.g.* [?]). We have access to an input set of m examples, also called prototypes, $\mathcal{S} \doteq \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m\}$. Vector $\boldsymbol{y}_i \in \{-1, 1\}^C$ encodes class memberships, assuming $y_{ic} = 1$

means that observation $\boldsymbol{x}_i$ belongs to class c. $\mathcal{S}$ is sampled i.i.d. according to an unknown distribution $\mathcal{D}$. Given an observation $\boldsymbol{x} \in \mathcal{O}$, we wish to estimate the conditional class probabilities for each class c, also called (estimated) posteriors:

$$\hat{p}_c(\boldsymbol{x}) \doteq \hat{\mathbf{Pr}}[y_c = 1|\boldsymbol{x}] \ . \tag{2}$$

We note $p_c(\boldsymbol{x}) \doteq \mathbf{Pr}_{\mathcal{D}}[y_c = 1|\boldsymbol{x}]$ the corresponding Bayes (true) posteriors.

2.2 Surrogates, losses and risks

Perhaps the simplest road towards computing these estimators consists in first crafting C separate classification problems, each of which leads to estimators for one class (2). Normalizing estimators to 1 over the C classes yields the values in (2). Each of these C problems is a one-versus-all classification task, say for class c, with corresponding sample $\mathcal{S}^{(c)} = \{(\boldsymbol{x}_i, y_{ic}), i = 1, 2, ..., m\}$. For each of these problems, we learn from $\mathcal{S}$ a classifier $h : \mathcal{O} \rightarrow \mathbb{R}$ out of which we may accurately compute (2), typically with $\hat{p}_c(\boldsymbol{x}) = f(h(\boldsymbol{x}))$ for some relevant function f. More sophisticated approaches exist that reduce the number of classifiers by folding classes in observation variables [**?**,**?**]. Each of them equivalently learn on a sample of $\Omega(mC)$ examples, and it is an easy task to craft from their output a set of C classifiers that fit into the framework we consider.

There exists a convenient approach to carry out this path as a whole, for each class $c = 1, 2, ..., C$: learn h by minimizing a *surrogate risk* over $\mathcal{S}$ [**?**,**?**,**?**]. A surrogate risk has general expression:

$$\varepsilon^{\psi}_{\mathcal{S}}(h, c) \doteq \frac{1}{m}\sum_{i=1}^{m} \psi(y_{ic}h(\boldsymbol{x})) \ , \tag{3}$$

for some function ψ that we call a *surrogate loss*. Quantity $y_{ic}h(\boldsymbol{x}) \in \mathbb{R}$ is called the *edge* of classifier $\boldsymbol{h}$ on example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ for class c. The surrogate risk is an estimator of the *true surrogate risk* computed over $\mathcal{D}$:

$$\varepsilon^{\psi}_{\mathcal{D}}(h, c) \doteq \mathbf{E}_{\mathcal{D}}[\psi(y_{ic}h(\boldsymbol{x}))] \ . \tag{4}$$

Any surrogate loss relevant to classification [**?**] has to meet $\mathrm{sign}(h_{\mathrm{opt}}(\boldsymbol{x}^*)) = \mathrm{sign}(2\mathbf{Pr}_{\mathcal{D}}[y_c = 1|\boldsymbol{x} = \boldsymbol{x}^*] - 1)$, where h_{opt} minimizes $\mathbf{E}_{\mathcal{D}}[\psi(y_c h(\boldsymbol{x}))|\boldsymbol{x} = \boldsymbol{x}^\star]$. Hence, the sign of the optimal classifier h_{opt} is as accurate to predict class membership as Bayes decision rule. This Fisher consistency requirement for ψ is called *classification calibration* [**?**]. We focus in this paper on the subclass of classification calibrated surrogates that are strictly convex and differentiable.

Definition 1. *[?] A* ***strictly convex loss*** *is a strictly convex function ψ differentiable on* $\mathrm{int}(\mathrm{dom}(\psi))$ *satisfying (i)* $\mathrm{im}(\psi) \subseteq \mathbb{R}^+$, *(ii)* $\mathrm{dom}(\psi)$ *symmetric around* 0, *(iii)* $\nabla_{\psi}(0) < 0$.

Definition 1 is extremely general: should we have removed conditions (i) and (ii), Theorem 6 in [**?**] brings that it would have encompassed the intersection between strictly convex differentiable functions and classification calibrated functions. Conditions (i) and (ii) are mainly conveniences for classification: in particular, it is not hard to see that modulo scaling by a positive constant, the surrogate risk (3) is an upperbound of the empirical risk for any strictly convex loss. Minimizing the surrogate risk amounts thus to minimize the empirical risk up to some extent. We define the Legendre conjugate of any strictly convex loss ψ as $\psi^{\star}(x) \doteq x\nabla_{\psi}^{-1}(x) - \psi(\nabla_{\psi}^{-1}(x))$. There exists a particular subset of strictly convex losses of independent interest [**?**]. A function $\phi : [0,1] \rightarrow \mathbb{R}^{+}$ is called *permissible* iff it is differentiable on $(0,1)$, strictly concave and symmetric around $x = 1/2$ [**?**,**?**]. We adopt the notation $\overline{\phi} = -\phi$ [**?**].

Definition 2. *[**?**] Given some permissible ϕ, we let ψ_{ϕ} denote the **balanced convex loss** with signature ϕ as:*

$$\psi_{\phi}(x) \doteq \frac{\overline{\phi}^{\star}(-x) - \phi(0)}{\phi\left(1/2\right) - \phi(0)} \ . \tag{5}$$

Balanced convex losses have an important rationale: up to differentiability constraints, they match the set of symmetric lower-bounded losses defining proper scoring rules [**?**], that is, basically, the set of losses that fit to classification problems without class-dependent misclassification costs. Table 1 provides examples of surrogate losses, most of which are strictly convex surrogates, some of which are balanced convex surrogates. We have derived Amari's α-loss from Amari's famed α divergences [**?**] (proof omitted). The linear Hinge loss is *not* a balanced convex loss, yet it figures the limit behavior of balanced convex losses [**?**]. Remark that all signatures ϕ are well-known in the domain of decision-tree induction : from the top-most to the bottom-most, one may recognize Gini criterion, the entropy (two expressions), Matsushita's criterion and the empirical risk [**?**,**?**].

2.3 One dimensional exponential families and posteriors estimation

A (regular) one dimensional *exponential family* [**?**] is a set of probability density functions whose elements admit the following canonical form:

$$p[x|\theta] \doteq \exp\left(x\theta - \psi(\theta)\right) p_0(x) \ , \tag{6}$$

where $p_0(x)$ normalizes the density, ψ is a strictly convex differentiable function that we call the *signature* of the family, and θ is the density's natural parameter. It was shown in [**?**] that the efficient minimization of any balanced convex surrogate risk — *i.e.* a surrogate risk with a balanced convex loss — amounts to a maximum likelihood estimation $\hat{\theta} = H(\boldsymbol{x})$ at some $\boldsymbol{x}$ for an exponential family whose signature depends solely on the permissible function ϕ. [**?**] suggest to use the corresponding *expected* parameter of the exponential family as the posterior:

$$\hat{\mathbf{Pr}}[y = 1|\boldsymbol{x}] = \hat{\mathbf{Pr}}_{\phi}[y = 1|\boldsymbol{x}; H] \doteq \nabla_{\overline{\phi}}^{-1}(H(\boldsymbol{x})) \in [0,1] \ . \tag{7}$$

$\nabla_{\overline{\phi}}^{-1}$ plays the role of the link function (1). The quality of such an estimator shall be addressed in the following Section.

	ψ	$\hat{p}_c(\boldsymbol{x})$	ϕ
A	$(1-x)^2$	$\frac{1}{2}(1+x)$	$x(1-x)$
B	$\log_2(1+\exp(-x))$	$[1+\exp(-x)]^{-1}$	$-x\ln x$ $-(1-x)\ln(1-x)$
C	$\log_2(1+2^{-x})$	$\left[1+2^{-x}\right]^{-1}$	$-x\log_2 x$ $-(1-x)\log_2(1-x)$
D	$-x+\sqrt{1+x^2}$	$\frac{1}{2}\left(1+\frac{x}{\sqrt{1+x^2}}\right)$	$\sqrt{x(1-x)}$
E	$\frac{1}{2}x(\mathrm{sign}(x)-1)$	$\begin{cases} 1 & \text{if} \quad x>0 \\ 0 & \text{if} \quad x<0 \end{cases}$	$2\min\{x, 1-x\}$
F	$\exp(-x)$	$[1+\exp(-2x)]^{-1}$	N/A
G	$\left(1+\frac{1-\alpha^2}{4}x\right)^{-\frac{1+\alpha}{1-\alpha}}$	$\left[1+\left(\frac{4-(1-\alpha^2)x}{4+(1-\alpha^2)x}\right)^{\frac{2}{1-\alpha}}\right]^{-1}$	N/A

Table 1: Examples of surrogates ψ (Throughout the paper, we let ln denote the base-e logarithm, and $\log_z(x) \doteq \ln(x)/\ln(z)$ denote the base-z logarithm). From top to bottom, the losses are known as: squared loss, (normalized) logistic loss, binary logistic loss, Matsushita loss [**?**,**?**], linear Hinge loss, exponential loss, Amari's α-loss, for $\alpha \in (-1,1)$ [**?**]. Strictly convex losses are A, B, C, D, F, G. Balanced convex losses are A, B, C, D (E corresponds to a limit behavior of balanced convex losses [**?**]). For each ψ, we give the corresponding estimators $\hat{p}_c(\boldsymbol{x})$ (Theorem 1 and Eqs (9, 11) below: replace x by $h_{\mathrm{opt}}(\boldsymbol{x})$), and if they are balanced convex losses, the corresponding concave signature ϕ (See text for details).

3 Strictly convex losses and the efficient estimation of posteriors

There is a rationale to use (7) as the posterior: the duality between natural and expectation parameters of exponential families, via Legendre duality [**?**,**?**], and the fact that the domain of the expectation parameter of one dimensional exponential families whose signature is (minus) a permissible function is the interval $[0,1]$ [**?**]. We improve below this rationale, with the proof that *Bayes posteriors* satisfy (7) for the classifier which is the population minimizer of (7).

Theorem 1. *Suppose ψ strictly convex differentiable. The true surrogate risk $\boldsymbol{E}_{\mathcal{D}}[\psi(y_{ic}h(\boldsymbol{x}))]$ is minimized at the unique $h_{\mathrm{opt}}(\boldsymbol{x})$ satisfying:*

$$\frac{\nabla_\psi(-h_{\mathrm{opt}}(\boldsymbol{x}))}{\nabla_\psi(h_{\mathrm{opt}}(\boldsymbol{x}))} = \frac{p_c(\boldsymbol{x})}{1-p_c(\boldsymbol{x})} \quad . \tag{8}$$

Furthermore, is ψ is a balanced convex loss, then the population minimizer h_{opt} of $\boldsymbol{E}_{\mathcal{D}}[\psi_\phi(y_{ic}h(\boldsymbol{x}))]$ satisfies:

$$p_c(\boldsymbol{x}) = \nabla_{\overline{\phi}}^{-1}(h_{\mathrm{opt}}(\boldsymbol{x})) \enspace , \tag{9}$$

for which

$$\boldsymbol{E}_{\mathcal{D}}[\psi_\phi(y_{ic}h_{\mathrm{opt}}(\boldsymbol{x}))] = \frac{\phi(p_c(\boldsymbol{x})) - \phi(0)}{\phi(1/2) - \phi(0)} \ . \tag{10}$$

(Proof omitted) Table 1 provides examples of expressions for $p_c(\boldsymbol{x})$ as in (9). Eq. (8) in Theorem (1) brings that we may compute an estimator $\hat{p}_c(\boldsymbol{x})$ as:

$$\hat{p}_c(\boldsymbol{x}) = \frac{\nabla_\psi(-h(\boldsymbol{x}))}{\nabla_\psi(h(\boldsymbol{x})) + \nabla_\psi(-h(\boldsymbol{x}))} \ . \tag{11}$$

This simple expression is folklore, at least for the logistic and exponential losses [**?**,**?**]. The essential contribution of Theorem 1 relies on bringing a strong rationale to the use of (7), as the estimators converge to Bayes posteriors in the infinite sample case. Let us give some finite sample properties for the estimation (7). We show that the sample-wise estimators of (9) are efficient estimators of (9); this is not a surprise, but comes from properties of exponential families [**?**]. What is perhaps more surprising is that the corresponding aggregation of classifiers is not a linear combination of all estimating classifiers, but a generalized $\nabla_{\overline{\phi}}^{-1}$-mean.

Theorem 2. *Suppose we sample n datasets $\mathcal{S}_j^{(c)}, j = 1, 2, ..., n$. Denote $\hat{h}_{\mathrm{opt},j}$ the population minimizer for $\boldsymbol{E}_{\mathcal{S}_j^{(c)}}[\psi_\phi(y_{ic}h(\boldsymbol{x}))]$. Then each $\hat{p}_{c,j}(\boldsymbol{x}) \doteq \nabla_{\overline{\phi}}^{-1}(\hat{h}_{\mathrm{opt},j}(\boldsymbol{x}))$ is the only efficient estimator for $p_c(\boldsymbol{x})$. The corresponding classifier $\hat{h}_{\mathrm{opt}}$ aggregating all $\hat{h}_{\mathrm{opt},j}$, is: $\hat{h}_{\mathrm{opt}}(\boldsymbol{x}) \doteq \nabla_{\overline{\phi}}\left(\frac{1}{n_{\boldsymbol{x}}}\sum_{j:(\boldsymbol{x},.)\in\mathcal{S}_j^{(c)}} \nabla_{\overline{\phi}}^{-1}(\hat{h}_{\mathrm{opt},j}(\boldsymbol{x}))\right), \forall \boldsymbol{x} \in \cup_j \mathcal{S}_j$, where $1 \leq n_{\boldsymbol{x}} \leq n$ is the number of subsets containing $\boldsymbol{x}$.*

Proof. Let us pick $\psi = \overline{\phi}^\star$ in (6) and condition $p[x|\theta] \doteq p[x|\theta; \boldsymbol{x}^*]$ for each $\boldsymbol{x}^* \in \mathcal{O}$. We let $\mu \doteq p_c(\boldsymbol{x}^*)$ (remark that $\mu \in \mathrm{dom}(\phi) = [0, 1]$ because ϕ is permissible) the expectation parameter of the exponential family, and thus $\theta = \nabla_{\overline{\phi}}(\mu)$. Using the fact that $\nabla_{\overline{\phi}^\star} = \nabla_{\overline{\phi}}^{-1}$, we get the score:

$$s(x|\theta) \doteq \frac{\partial \ln p[x|\theta]}{\partial \theta} = x - \nabla_{\overline{\phi}^\star}(\theta) \ ,$$

and so x is an efficient estimator for $\nabla_{\overline{\phi}^\star}(\theta) = \mu$; in fact, it is the only efficient estimator [**?**]. Thus, $\hat{p}_c(\boldsymbol{x}^*)$ is an efficient estimator for $p_c(\boldsymbol{x}^*)$. There remains to use (9) to complete the proof of Theorem 2. □

4 Leveraging and boosting Nearest Neighbors

The nearest neighbor rule belongs to the oldest, simplest and most widely studied classification algorithms [**?**,**?**]. We denote by $\mathrm{NN}_k(\boldsymbol{x})$ the set of the k-nearest neighbors (with integer constant $k > 0$) of an example $(\boldsymbol{x}, \boldsymbol{y})$ in set $\mathcal{S}$ with respect to a non-negative real-valued "distance" function. This function is defined on

Algorithm 1 Algorithm UNIVERSAL NEAREST NEIGHBORS, UNN$(\mathcal{S}, \psi, k)$

```
Input: $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m,\ \boldsymbol{x_i} \in \mathcal{O},\ \boldsymbol{y_i} \in \{-1, 1\}^C\}$, $\psi$ strictly convex loss
       (Definition 1), $k \in \mathbb{N}_*$;
Let $\boldsymbol{\alpha}_j \leftarrow \boldsymbol{0}, \forall j = 1, 2, ..., m$;
for $c = 1, 2, ..., C$ do
    Let $\boldsymbol{w} \leftarrow -\nabla_\psi(0)\boldsymbol{1}$;
    for $t = 1, 2, ..., T$ do
        [I.0] Let $j \leftarrow$ WIC$(\mathcal{S}, \boldsymbol{w})$;
        [I.1] Let $\delta_j \in \mathbb{R}$ solution of:
              $$\sum_{i : j \sim_k i} y_{ic} y_{jc} \nabla_\psi \left(\delta_j y_{ic} y_{jc} + \nabla_\psi^{-1}(-w_i)\right) = 0 \ ; \qquad (12)$$
        [I.2] $\forall i : j \sim_k i$, let
              $$w_i \leftarrow -\nabla_\psi \left(\delta_j y_{ic} y_{jc} + \nabla_\psi^{-1}(-w_i)\right) \ , \qquad (13)$$
        [I.3] Let $\alpha_{jc} \leftarrow \alpha_{jc} + \delta_j$;
Output: $\mathcal{H}(\boldsymbol{x}) \doteq \sum_{j \sim_k \boldsymbol{x}} \boldsymbol{\alpha}_j \circ \boldsymbol{y}_j$
```

domain $\mathcal{O}$ and measures how much two observations differ from each other. This dissimilarity function thus may not necessarily satisfy the triangle inequality of metrics. For the sake of readability, we let $j \sim_k \boldsymbol{x}$ denote the assertion that example $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ belongs to $\mathrm{NN}_k(\boldsymbol{x})$. We shall abbreviate $j \sim_k \boldsymbol{x}_i$ by $j \sim_k i$. To classify an observation $\boldsymbol{x} \in \mathcal{O}$, the k-NN rule $\mathcal{H}$ over $\mathcal{S}$ computes the sum of class vectors of its nearest neighbors, that is: $\mathcal{H}(\boldsymbol{x}) = \sum_{j \sim_k \boldsymbol{x}} \boldsymbol{1} \circ \boldsymbol{y}_j$, where $\circ$ is the Hadamard product. $\mathcal{H}$ predicts that $\boldsymbol{x}$ belongs to each class whose corresponding coordinate in the final vector is positive. A *leveraged* k-NN rule is a generalization of this to:

$$\mathcal{H}(\boldsymbol{x}) = \sum_{j \sim_k \boldsymbol{x}} \boldsymbol{\alpha}_j \circ \boldsymbol{y}_j \ , \tag{14}$$

where $\boldsymbol{\alpha}_j \in \mathbb{R}^C$ is a leveraging vector for the classes in $\boldsymbol{y}_j$. Leveraging approaches to nearest neighbors are not new [**?**,**?**], yet to the best of our knowledge no convergence results or rates were known, at least until the algorithm UNN [**?**]. Algorithm 1 gives a simplified version of the UNN algorithm of [**?**] which learns a leveraged k-NN. Oracle WIC$(\mathcal{S}, \boldsymbol{w})$ is the analogous for NN of the classical weak learners for boosting: it takes learning sample $\mathcal{S}$ and weights $\boldsymbol{w}$ over $\mathcal{S}$, and returns the index of some example in $\mathcal{S}$ which is to be leveraged. [**?**] prove that for any strictly convex loss ψ, UNN converges to the global optimum of the surrogate risk at hand. However, they prove boosting-compliant convergence rates only for the exponential loss. For all other strictly convex losses, there is no insight on the rates with which UNN may converge towards the optimum of the surrogate risk at hand. We now provide such explicit convergence rates under the following *Weak Learning Assumption*:

WLA: There exist some $\vartheta > 0, \varrho > 0$ such that, given any $k \in \mathbb{N}_*$, $c = 1, 2, ..., C$ and any distribution $\boldsymbol{w}$ over $\mathcal{S}$, the weak index chooser oracle Wic returns an index j such that the following two statements hold:

(i) $\mathbf{Pr}_{\boldsymbol{w}}[j \sim_k i] \geq \varrho$;

(ii) $\mathbf{Pr}_{\boldsymbol{w}}[y_{jc} \neq y_{ic} | j \sim_k i] \leq 1/2 - \vartheta$ or $\mathbf{Pr}_{\boldsymbol{w}}[y_{jc} \neq y_{ic} | j \sim_k i] \geq 1/2 + \vartheta$.

Requirement (i) is a weak *coverage* requirement, which "encourages" Wic to choose indexes in dense regions of $\mathcal{S}$. Before studying the boosting abilities of unn, we focus again on surrogate risks. So far, the surrogate risk (3) has been evaluated with respect to a single class. In a multiclass multilabel setting, we may compute the *total* surrogate risk over all classes as:

$$\varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H}) \doteq \frac{1}{C} \sum_{c=1}^{C} \varepsilon^{\psi}_{\mathcal{S}}(h_c, c) \ , \tag{15}$$

where $\mathcal{H}$ is the set of all C classifiers $h_1, h_2, ..., h_C$ that have been trained to minimize each $\varepsilon^{\psi}_{\mathcal{S}}(., c), c = 1, 2, ..., C$. We split classifiers just for convenience in the analysis: if one trains a single classifier $H : \mathcal{O} \times \{1, 2, ..., C\} \rightarrow \mathbb{R}$ like for example [**?**], then we define h_c to be H in which the second input coordinate is fixed to be c. Minimizing the total surrogate risk is not only efficient to estimate posteriors (Section 3): it is also useful to reduce the error in label prediction, as the total surrogate risk is an upperbound for the *Hamming risk* [**?**]: $\varepsilon^{\mathrm{H}}_{\mathcal{S}}(\mathcal{H}) \doteq (1/(mC)) \sum_{c=1}^{C} \sum_{i=1}^{m} \mathrm{I}[y_{ic} h_c(\boldsymbol{x}_i) < 0]$, where $\mathrm{I}[.]$ denotes the indicator variable. It is indeed not hard to check that for any strictly convex surrogate loss ψ, we have $\varepsilon^{\mathrm{H}}_{\mathcal{S}}(\mathcal{H}) \leq (1/\psi(0)) \times \varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H})$. We are left with the following question about unn:

"are there sufficient conditions on the surrogate loss ψ that guarantee, under the sole **WLA**, a *convergence rate* towards the optimum of (15) with unn ?"

We give a positive answer to this question when the surrogate loss meets the following smoothness requirement.

Definition 3. *[**?**] ψ is said to be ω strongly smooth iff there exists some $\omega > 0$ such that, for all $x, x' \in \mathrm{int}(\mathrm{dom}(\psi))$, $D_{\psi}(x' \| x) \leq \frac{\omega}{2}(x' - x)^2$, where*

$$D_{\psi}(x' \| x) \doteq \psi(x') - \psi(x) - (x' - x)\nabla_{\psi}(x) \tag{16}$$

*denotes the Bregman divergence with generator ψ [**?**].*

Denote $n_j \doteq |\{i : j \sim_k i\}|$ the number of examples in $\mathcal{S}$ of which $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ is a nearest neighbor, and $n_* \doteq \max_j n_j$. Denote also $\mathcal{H}_{\mathrm{opt}}$ the leveraged k-nn which minimizes $\varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H})$; it corresponds to the set of classifiers $\hat{h}_{\mathrm{opt}}$ of Section 3 that would minimize (3) over each class. We are now ready to state our main result (remark that $\varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H}_{\mathrm{opt}}) \leq \psi(0)$).

Theorem 3. *Suppose (**WLA**) holds and choose as ψ is any ω strongly smooth, strictly convex loss. Then for any fixed $\tau \in [\varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H}_{\mathrm{opt}}), \psi(0)]$,* UNN *has fit a leveraged k-*NN *classifier $\mathcal{H}$ satisfying $\varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H}) \leq \tau$ provided the number of boosting iterations T in the inner loop satisfies:*

$$T \geq \frac{(\psi(0) - \tau)\omega m n_*}{2\vartheta^2 \varrho^2} \ . \tag{17}$$

Proof sketch: To fit UNN to the notations of (15), we let h_c represent the leveraged k-NN in which each $\boldsymbol{\alpha}_j$ is restricted to α_{jc}. We first analyze $\varepsilon^{\psi}_{\mathcal{S}}(h_c, c)$ for some fixed c in the outer loop of Algorithm 1, after all α_{jc} have been computed in the inner loop. We adopt the following notations in this proof: we plug in the weight notation the iteration t and class c, so that $w^{(c)}_{ti}$ denotes the weight of example $\boldsymbol{x}_i$ at the beginning of the "**for** c" loop of Algorithm 1.

ψ is ω strongly smooth is equivalent to $\tilde{\psi}$ being strongly convex with parameter ω^{-1} [**?**], that is,

$$\tilde{\psi}(w) - \frac{1}{2\omega} w^2 \text{ is convex,} \tag{18}$$

where we use notation $\tilde{\psi}(x) \doteq \psi^{\star}(-x)$. Any convex function h satisfies $h(w') \geq h(w) + \nabla_h(w)(w' - w)$. We apply this inequality taking as h the function in (18). We obtain, $\forall t = 1, 2, ..., T, \forall i = 1, 2, ..., m, \forall c = 1, 2, ..., C$:

$$D_{\tilde{\psi}}\left(w^{(c)}_{(t+1)i} || w^{(c)}_{ti}\right) \geq \frac{1}{2\omega}\left(w^{(c)}_{(t+1)i} - w^{(c)}_{ti}\right)^2 \ . \tag{19}$$

On the other hand, Cauchy-Schwartz inequality and (12) yield:

$$\forall j \in \mathcal{S}, \sum_{i:j\sim_k i}\left(\mathrm{r}^{(c)}_{ij}\right)^2 \sum_{i:j\sim_k i}(w^{(c)}_{(t+1)i} - w^{(c)}_{ti})^2 \geq \left(\sum_{i:j\sim_k i}\mathrm{r}^{(c)}_{ij} w^{(c)}_{ti}\right)^2 \ . \tag{20}$$

Lemma 1. *Under the **WLA**, index j returned by* WIC *at iteration t satisfies $\left|\sum_{i:j\sim_k i} w^{(c)}_{ti}\mathrm{r}^{(c)}_{ij}\right| \geq 2\vartheta\varrho$.*

(proof omitted) Letting $e(t) \in \{1, 2, ..., m\}$ denote the index of the example returned at iteration t by WIC in Algorithm 1, we obtain:

$$\frac{1}{m}\sum_{i=1}^{m} D_{\tilde{\psi}}\left(w^{(c)}_{(t+1)i} || w^{(c)}_{ti}\right) \geq \frac{1}{2\omega m}\sum_{i:e(t)\sim_k i}\left(w^{(c)}_{(t+1)i} - w^{(c)}_{ti}\right)^2 \tag{21}$$

$$\geq \frac{1}{2\omega m}\frac{\left(\sum_{i:e(t)\sim_k i}\mathrm{r}^{(c)}_{ie(t)} w^{(c)}_{ti}\right)^2}{\sum_{i:e(t)\sim_k i}\left(\mathrm{r}^{(c)}_{ie(t)}\right)^2} \tag{22}$$

$$\geq \frac{2\vartheta^2\varrho^2}{\omega m} \times \frac{1}{\sum_{i:e(t)\sim_k i}\left(\mathrm{r}^{(c)}_{ie(t)}\right)^2} \tag{23}$$

$$= \frac{2\vartheta^2\varrho^2}{\omega m n_{e(t)}} \geq \frac{2\vartheta^2\varrho^2}{\omega m n_*} \ . \tag{24}$$

Here, (21) follows from (19), (22) follows from (20), (23) follows from Lemma 1, and (24) follows from the fact that $\mathrm{r}^{(c)}_{ie(t)} = \pm 1$ when $e(t) \sim_k i$. Summing these inequalities for $t = 1, 2, ..., T$ yields:

$$\sum_{t=1}^{T} \frac{1}{m} \sum_{i=1}^{m} D_{\tilde{\psi}} \left(w^{(c)}_{(t+1)i} || w^{(c)}_{ti} \right) \geq \frac{2T\vartheta^2 \varrho^2}{\omega m n_*} \ . \tag{25}$$

Now, UNN meets the following property ([**?**], A.2):

$$\varepsilon^{\psi}_{\mathcal{S}}(h_{(t+1)c}, c) - \varepsilon^{\psi}_{\mathcal{S}}(h_{tc}, c) = -\frac{1}{m} \sum_{i=1}^{m} D_{\tilde{\psi}} \left(w^{(c)}_{(t+1)i} || w^{(c)}_{ti} \right), \tag{26}$$

where $h_{(t+1)c}$ denotes h_c after the t^{th} iteration in the inner loop of Algorithm 1. We unravel (26), using the fact that all $\boldsymbol{\alpha}$ are initialized to the null vector, and obtain that at the end of the inner loop, h_c satisfies:

$$\varepsilon^{\psi}_{\mathcal{S}}(h_c, c) = \psi(0) - \sum_{t=1}^{T} \frac{1}{m} \sum_{i=1}^{m} D_{\tilde{\psi}} \left(w^{(c)}_{(t+1)i} || w^{(c)}_{ti} \right) \leq \psi(0) - \frac{2T\vartheta^2 \varrho^2}{\omega m n_*} \ , \tag{27}$$

from (25). There remains to compute the minimal value of T for which the right hand side of (27) becomes no greater than some user-fixed $\tau \in [0, 1]$ to obtain that $\varepsilon^{\psi}_{\mathcal{S}}(h_c, c) \leq \tau$.

The aggregation of the bounds for each $c = 1, 2, ..., C$ in $\varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H})$ is immediate as it is an average of $\varepsilon^{\psi}_{\mathcal{S}}(h_c, c)$ over all classes. Hence, this minimal value of T, used for each $c = 1, 2, ..., C$, also yields $\varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H}) \leq \tau$. This ends the proof of Theorem 3. □

Section 3 has underlined the importance of balanced convex losses in obtaining simple efficient estimators for conditional class probabilities. Coupled with Theorem 3, we now show that UNN may be a fast approach to obtain such estimators.

Corollary 1. *Consider any permissible ϕ that has been scaled without loss of generality so that $\phi(1/2) = 1$, $\phi(0) = \phi(1) = 0$. Then for the corresponding balanced convex loss $\psi = \psi_\phi$ and under the **WLA**, picking*

$$T > \frac{m n_*}{2\vartheta^2 \varrho^2 \min_{x \in (0,1)} \left| \frac{\partial^2 \phi}{\partial x^2} \right|} \tag{28}$$

in the inner loop of UNN, for each $c = 1, 2, ..., C$, guarantees to yield an optimal leveraged k-NN $\mathcal{H}$, satisfying $\varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H}) = \varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H}_{\mathrm{opt}})$. This leveraged k-NN yields efficient estimators for conditional class probabilities, for each class, by computing:

$$\hat{p}_c(\boldsymbol{x}) = \nabla^{-1}_{\tilde{\phi}}(h_c(\boldsymbol{x})) \ . \tag{29}$$

(Proof omitted) For the most popular permissible functions (Table 1), quantity $\min_{x \in (0,1)} \left| \frac{\partial^2 \phi}{\partial x^2} \right|$ does not take too small value: its values are respectively 8, $4/\ln 2$, 4 for the permissible functions corresponding to the squared loss, logistic loss, Matsushita loss. Hence, in these cases, the bound for T in (28) is not significantly affected by this term.

	δ_{jc}, see (30)	$g : w_i \leftarrow g(w_i)$
A	$2W_{jc} - 1$	$w_i - 2\delta_{jc}y_{ic}y_{jc}$
B	$\ln \frac{W_{jc}}{1-W_{jc}}$	$\frac{w_i}{w_i \ln 2 + (1-w_i \ln 2) \times \exp(\delta_{jc}y_{ic}y_{jc})}$
C	$\log_2 \frac{W_{jc}}{1-W_{jc}}$	$\frac{w_i}{w_i + (1-w_i) \times 2^{\delta_{jc}y_{ic}y_{jc}}}$
D	$\frac{2W_{jc}-1}{2\sqrt{W_{jc}(1-W_{jc})}}$	$1 - \frac{1-w_i+\sqrt{w_i(2-w_i)}\delta_{jc}y_{ic}y_{jc}}{\sqrt{1+\delta_{jc}^2 w_i(2-w_i)+2(1-w_i)\sqrt{w_i(2-w_i)}\delta_{jc}y_{ic}y_{jc}}}$
E	N/A	N/A
F	$\frac{1}{2} \ln \frac{W_{jc}}{1-W_{jc}}$	$\exp(-\delta_{jc}y_{ic}y_{jc})$
G	$\frac{4}{1-\alpha^2} \left(\frac{(W_{jc})^{\frac{2}{1-\alpha}} - (1-W_{jc})^{\frac{2}{1-\alpha}}}{(W_{jc})^{\frac{2}{1-\alpha}} + (1-W_{jc})^{\frac{2}{1-\alpha}}} \right)$	$\frac{4}{1-\alpha^2} \times \left(\frac{1-\alpha^2}{4}\delta_{jc}y_{ic}y_{jc} + \left(\frac{1+\alpha}{2\sqrt{w_i}}\right)^{1-\alpha} \right)^{-\frac{2}{1-\alpha}}$

Table 2: Computation of δ_{jc} and the weight update rule of our implementation of UNN, for the strictly convex losses of Table 1. UNN leverages example j for class c, and the weight update is that of example i (See text for details and notations).

5 Experiments

5.1 Computing leveraging coefficients and weights update

Fix for short $\mathcal{S}_{jb}^{(c)} \doteq \{i : j \sim_k i \wedge y_{ic} = by_{jc}\}$ for $b \in \{+, -\}$. (12) may be simplified as $\sum_{i \in \mathcal{S}_{j+}^{(c)}} \nabla_\psi \left(\delta + \nabla_\psi^{-1}(-w_i)\right) = \sum_{i \in \mathcal{S}_{j-}^{(c)}} \nabla_\psi \left(-\delta + \nabla_\psi^{-1}(-w_i)\right)$. There is no closed form solution to this equation in the general case. While it can be simply approximated with dichotomic search, it buys significant computation time, as this approximation has to be performed for each couple (c, t). We tested a much faster alternative which produces results that are in general experimentally quite competitive, consisting in solving instead: $\sum_{i \in \mathcal{S}_{j+}^{(c)}} w_i \nabla_\psi(\delta) = \sum_{i \in \mathcal{S}_{j-}^{(c)}} w_i \nabla_\psi(-\delta)$. We get equivalently that δ satisfies:

$$\frac{\nabla_\psi(-\delta)}{\nabla_\psi(\delta)} = \frac{W_{jc}}{1 - W_{jc}} \ , \tag{30}$$

with $W_{jc} \doteq (\sum_{i \in \mathcal{S}_{j+}^{(c)}} w_i)/(\sum_{i \in \mathcal{S}_{j+}^{(c)}} w_i + \sum_{i \in \mathcal{S}_{j-}^{(c)}} w_i)$. Remark the similarity with (8). Table 2 gives the corresponding expressions for δ and the weight updates.

5.2 General experimental settings

We have tested three flavors of UNN: with the exponential loss (F in Table 1), the logistic loss (B in Table 1) and Matsushita's loss (D in Table 1). All three a respectively referred to as UNN(exp), UNN(log) and UNN(Mat). It is the first time this last flavor is tested, even from the classification standpoint. We chose support vector machines (SVM) as the contender against which to compare UNN:

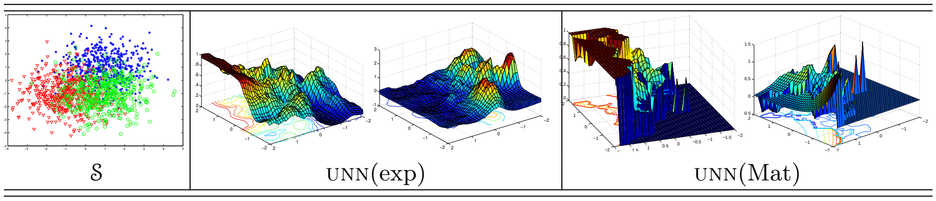

Fig. 1: From left to right: example of simulated dataset with $\sigma = 1.1$; the estimated posterior for class 1 obtained by UNN(exp); the corresponding gridwise KL divergence for class 1; the estimated posterior for class 1 obtained by UNN(Mat); the corresponding gridwise KL divergence for class 1 (see (32) and text for details).

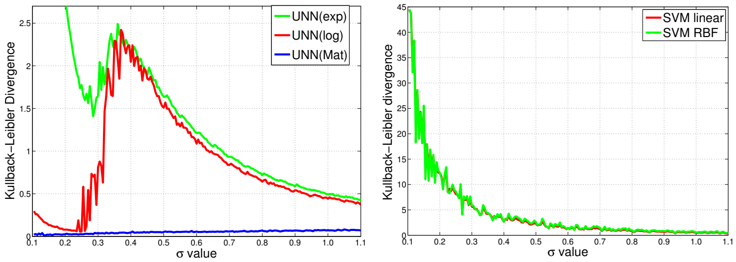

Fig. 2: Average KL-divergence as a function of σ on simulated datasets, for UNN(exp), UNN(log), UNN(Mat) (left, $k = 10$) and SVM (right). Notice the differences in y-scales.

SVM are large margin classifiers with convenient methods to obtain estimators for the posteriors [**?**]. For all these algorithms, we compute the estimation of posteriors as follows: we use (11) for UNN(exp), (29) for UNN(log) and UNN(Mat). For SVM, we use the method of [**?**], which, given a SVM f for class c, forms the posterior:

$$\hat{p}_c(\boldsymbol{x}) \doteq \frac{1}{1+\exp(af(\boldsymbol{x})+b)} \ , \tag{31}$$

where a and b are estimated by maximizing the log-likelihood of the training sample with a five-fold cross validation. We use two metrics to evaluate the algorithms. On simulated data, we compute an estimate of the Kullback-Leibler (KL) divergence between the true and estimated posterior which is a class-wise average of the divergence:

$$D_{\mathrm{KL}}(\hat{p}\|p) \doteq \sum_c \mathbf{Pr}[c] \int \mathbf{Pr}[\boldsymbol{x}]\hat{p}_c(\boldsymbol{x}) \ln \frac{\hat{p}_c(\boldsymbol{x})}{p_c(\boldsymbol{x})} \mathrm{d}\mu \ . \tag{32}$$

Our estimate, $\hat{D}_{\mathrm{KL}}(\hat{p}\|p)$ relies on a simple fine-grained grid approximation of the integral over the subsets of $\mathcal{O}$ of sufficient mass according to μ. On real

	k	UNN(exp)	UNN(log)	UNN(Mat)	SVM$_l$	SVM$_r$
$\hat{D}_{\mathrm{KL}}(\hat{p} \| p)$	10	1.649	0.862	0.052	4.303	4.379
	20	0.721	0.651	0.038		
	30	0.589	0.534	0.034		
	40	0.523	0.492	0.033		
F-measure	10	90.32	89.59	90.58	91.02	90.90
	20	90.62	89.53	90.81		
	30	90.70	89.26	90.84		
	40	90.72	88.82	90.88		

Table 3: Average results over simulated data, for UNN(exp), UNN(log), UNN(Mat) with four different values of k, and for support vector machines with linear (SVM$_l$) or radial basis functions (SVM$_r$) kernel.

data, we compute a couple of metrics. First, we compute the F-measure of the classifiers (the harmonic average of precision and recall), based on thresholding the probabilistic output and deciding that $\boldsymbol{x}$ belong to class c iff $\hat{p}_c(\boldsymbol{x}) \geq \kappa$, for varying $\kappa \in (1/2, 1)$. Second, we compute the rejection rate, that is, the proportion of observations for which $\hat{p}_c(\boldsymbol{x}) < \kappa$. Either we plot couples of curves for the F-measure and rejection rates, or we summarize both metrics by their average values as κ ranges through $(1/2, 1)$, which amounts to compute the area under the corresponding curves.

5.3 Results on simulated data

We evaluated the goodness-of-fit of the estimates on simulated datasets with the following experiments. We crafted a general domain consisting of $C = 3$ equiprobable classes, each of which follows a Gaussian $\mathcal{N}(\boldsymbol{\mu}, \sigma \mathrm{I})$, for $\sigma \in [0.1, 1.1]$ with steps of 0.005, and $\boldsymbol{\mu}$ remains the same. For each value of σ, we compute the average over ten simulations, each of which consists of 1500 training examples and 4500 testing examples. We get overall several thousands datasets, on which all algorithms are tested. Figure 1 presents an example of such datasets, along with results obtained by UNN(exp) and UNN(Mat) from the standpoints of the posterior estimates and KL-divergence on the same class. The estimators are rather good, with the largest mismatches (KL-divergence) located near the frontiers of classes. Also, UNN(Mat) tends to outperform UNN(exp).

Figure 2 synthesizes the results from the KL-divergence standpoints. Two clear conclusions can be drawn from these results. First, UNN is the clear winner over SVM for the posteriors estimation task. The results of each flavor of UNN is indeed better than those of SVM, with linear or radial basis functions kernel, by orders of magnitude. This is all the more important as the kernels we used are the theoretical kernels of choice given the way we have simulated data. The second conclusion is that UNN(Mat) is the best of all flavors of UNN, a fact also confirmed by the synthetic results of Table 3. The KL divergences of UNN(Mat) are in general of minute order with respect to the others. Its behavior (Figure 2) is also monotonous: it is predictable that it increases with the degree of overlap between classes, that is, with σ. From the *classification* standpoint, the average F-measure

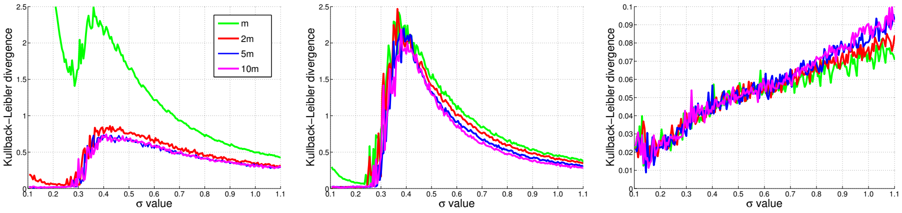

Fig. 3: Average KL-divergence as a function of σ on simulated datasets, for UNN(exp) (left), UNN(log) (center), UNN(Mat) (right), when the number of boosting iterations T varies in $\{m, 2m, 5m, 10m\}$. The color code in the same on each plot. Notice the differences in the y-scale for UNN(Mat) (see text for details).

metrics display a very slight advantage to SVM, and in particular to linear SVM. The results of SVM with radial basis functions kernel are approximately the same as those of UNN(Mat).

The most important conclusion that can be drawn from the simulated data is shown in Figure 3: as the number of boosting iterations T increase, UNN does *not* overfit posteriors in general. The only hitch — not statistically significant — is the case $\sigma > 0.7$ for UNN(Mat), but the differences are of very small order compared to the standard deviations of the KL-divergence.

5.4 Results on the SUN database domains

	UNN(exp)		UNN(log)		UNN(Mat)		SVM$_l$	
	F	R	F	R	F	R	F	R
SUN 10	**89.91**	21.35	84.46	5.18	72.47	**3.39**	87.99	22.32
SUN 20	**82.82**	36.64	72.34	8.51	55.46	**2.51**	74.60	33.25
SUN 30	**73.39**	49.92	61.02	14.99	40.83	**5.99**	62.81	39.95

Table 4: Area under the (F)-measure (in percentage) and (R)ejection rate on the SUN databases. For each database, the best F and R are written in **bold faces**.

We have crafted, out of the challenging SUN computer vision database [**?**], three datasets, consisting in taking all pictures from the first ten (SUN 10), twenty (SUN 20) or thirty (SUN 30) classes. We have compared UNN(exp), UNN(log), UNN(Mat) and SVM on each dataset, by computing the average values, over the threshold κ, of the F-measure and the rejection rate. Table 4 summarizes the results obtained. This table somehow confirms that classification and posterior estimation may be conflicting goals when it comes to boosting [**?**,**?**], as UNN(Mat) achieves very poor results compared to the other algorithms. Furthermore, UNN(exp) appears to the clear winner over all algorithms for this classification task. These results have to be appreciated in the light of the rejection rates: in comparison with the other algorithms, UNN(Mat) rejects a very

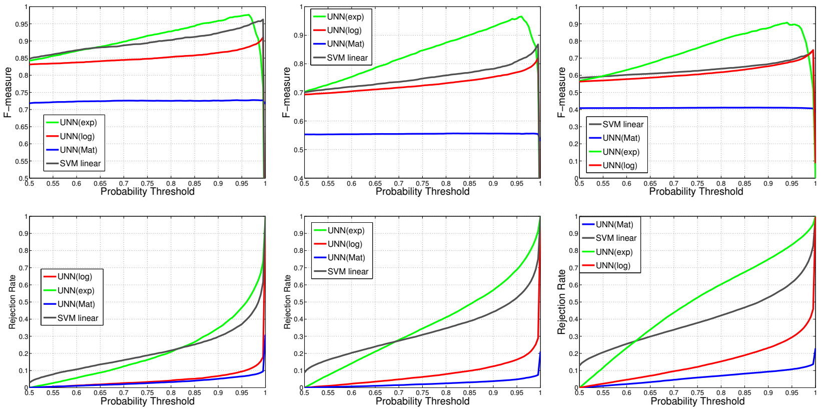

Fig. 4: F-measure (top row) and rejection rates (bottom row) on the SUN domains, with $C = 10$ (left), $C = 20$ (center) and $C = 30$ (right, see Table 3 for notations).

small proportion of the examples, this indicating a high recall for the algorithm. Figure 4 completes the picture by detailing F-measure and rejection rates plots. The F-measure plots clearly display the better performances of UNN(exp) compared to the other algorithms, and the fact that UNN(Mat) displays very stable performances. The rejection rates plots show that UNN(Mat) indeed rejects a very small proportion of examples, even for large values of κ.

6 Conclusion

Boosting algorithms are remarkably simple and efficient from the classification standpoint, and are being used in a rapidly increasing number of domains and problems [**?**]. In some sense, it would be too bad that such successes be impeded when it comes to posterior estimation [**?**]. Experimental results display that this estimation is possible, but it necessitates a very fine tuning of the algorithms [**?**]. The point of our paper is that estimating class conditional probabilities may be possible, without such tedious tunings, and sometimes even *without overfitting*, if we boost topological approaches to learning like nearest neighbors. There is a simple explanation to this fact. For any classifier, the conditional class probability estimation for some $\boldsymbol{x}$ in (7) is be the same as for any other observation in the vicinity of $\boldsymbol{x}$, where the "vicinity" is to be understood from the *classifier* standpoint. When boosting decision trees, the vicinity of $\boldsymbol{x}$ corresponds to observations classified by the same leaf as $\boldsymbol{x}$. As the number of leaves of the tree increases, the vicinity gets narrowed, which weakens the estimation in (7) and thus overfits the corresponding estimated density. Ultimately, linear combinations of such trees, such as those performed in AdaBoost, make such a fine-grained approximation of the local topology of data that the estimators get irreparably confined to the borders of the interval $[0, 1]$ [**?**]. Nearest neighbors do not have such a drawback, as the set of k-nearest neighbors in $\mathcal{S}$ of some observation $\boldsymbol{x}$ spans a region of $\mathcal{O}$ which does not change throughout the iterations. Furthermore, nearest neighbor rules exploit a topology of data which, under regularity conditions about the true posteriors, also carries out information about these posteriors. For these reasons, nearest neighbors might be a key entry for a reliable estimation of posteriors with boosting. Because of the wealth of "good" surrogates, this opens avenues of research to *learn* the most accurate surrogate on a data-dependent way, such as when it is parameterized (Amari's α-loss, see Table 1).

7 Acknowledgments

R. Nock acknowledges a visiting grant from Institut Universitaire de France / Université de Nice.