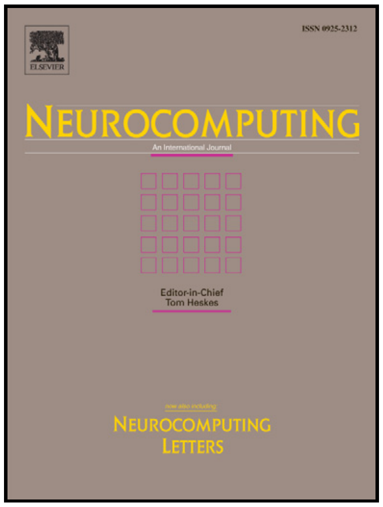


www.elsevier.com/locate/neucom

Leveraging k-NN for generic classification boosting

Paolo Piro, Richard Nock, Frank Nielsen, Michel Barlaud

PII: S0925-2312(11)00598-4
DOI: doi:10.1016/j.neucom.2011.07.026
Reference: NEUCOM 12411

To appear in: *Neurocomputing*

Received date: 17 March 2011
Revised date: 1 July 2011
Accepted date: 28 July 2011

Leveraging k-NN for generic classification boosting

Paolo Piro[a,d], Richard Nock[b], Frank Nielsen[c], Michel Barlaud[a]

[a]*CNRS/University of Nice-Sophia Antipolis*
[b]*CEREGMIA, University of Antilles-Guyane, France*
[c]*LIX, Ecole Polytechnique, France*
[d]*Italian Institute of Technology, Italy*

Abstract

Voting rules relying on k-nearest neighbors (k-NN) are an effective tool in countless many machine learning techniques. Thanks to its simplicity, k-NN classification is very attractive to practitioners, as it enables very good performances in several practical applications. However, it suffers from various drawbacks, like sensitivity to "noisy" instances and poor generalization properties when dealing with sparse high-dimensional data.
In this paper, we tackle the k-NN classification problem at its core by providing a novel k-NN boosting approach. Namely, we propose a supervised learning algorithm, called *Universal Nearest Neighbors* (UNN), that induces a leveraged k-NN rule by globally minimizing a *surrogate risk* upper bounding the empirical misclassification rate over training data. Interestingly, this surrogate risk can be arbitrary chosen from a class of Bregman loss functions, including the familiar exponential, logistic and squared losses. Furthermore, we show that UNN allows to efficiently filter a dataset of instances by keeping only a small fraction of data.
Experimental results on the synthetic Ripley's dataset show that such a filtering strategy is able to reject "noisy" examples, and yields a classification error close to the optimal Bayes error. Experiments on standard UCI datasets show significant improvements over the current state of the art.

Keywords: Boosting, k-NN classification, surrogate risks.

1. Introduction

k-NN classification is nowadays a crucial tool widely used in several areas of computer science, like pattern recognition, computer vision and data mining [1]. Namely, in the context of machine learning, nearest neighbor methods have been applied to a variety of supervised problems. In spite of its simplicity and the lack of theoretical guarantees for training sets of finite size [2], k-NN classifiers have been observed to perform very well in practice, often even outperfoming far more sophisticated techniques. However, it is an endeavor algorithmic challenge to speed-up k-NN queries and design schemes that scale up well with large dimensional datasets. Moreover, reducing the classification error of the k-NN rule [1] is yet another crucial challenge, relying on two main issues that may significantly influence the performances. The first one is related to the similarity criterion, which depends on the underlying distance measure as well as on the selection criterion, *e.g.*, fixing the value of k. This problem has been often addressed by learning an appropriate metric in the feature space, *e.g.*, exploiting pairwise distance contraints between training points [3]. The second issue is how to combine the labels of the neighbors, *i.e.*, which voting rule to use for predicting unknown classes. The simple uniform majority vote may penalize k-NN classification, *e.g.*, when dealing with high-dimensional data and a large number of classes. This issue has been usually tackled by data reduction techniques [4].

Furthermore, in a number of works, the classification problem has been reduced to tracking ill-defined categories of neighbors, interpreted as "noisy" [5]. Most of these recent techniques are in fact partial solutions solving a larger problem related to nearest neighbors' error, which does not have to be the discrete prediction of labels, but rather a continuous estimation of class membership probabilities [6]. This problem has been reformulated by Marin et al [7] as a strong advocacy for the formal transposition of *boosting* to nearest neighbor classification. This is challenging, as nearest neighbor rules are indeed not *induced*, while all formal boosting algorithms induce so-called *strong* classifiers by combining *weak* (also induced) classifiers. (Along the paper, the word "induction" is used in a statistically-inferential sense, as originally proposed by Quinlan [8].)

In this paper, we describe a novel solution to the problem of k-NN boosting in its most general setting, *i.e.*, multiclass classification. In particular, we propose a *Universal Nearest Neighbors* (UNN) algorithm, which *induces* a *leveraged* k-nearest neighbor rule generalizing the uniform k-NN rule. We show that UNN converges to the *global* minimum of any chosen *classification calibrated surrogate*[1]. Our framework thus handles most popular losses

[1]A *surrogate* is a function that is a suitable upperbound for

already met in the literature: exponential loss, logistic loss, squared loss, etc. We prove a specific convergence rate for the exponential loss (used in our experiments) far better than the general rate. We validated UNN on the synthetic Ripley's dataset [10] and several standard UCI datasets. Experiments display significant accuracy improvements with UNN over classic k-NN classification, with the additional advantage of considerably reducing the classification time. Furthermore, our algorithm often outperforms a state-of-the-art method relying on k-NN metric learning.

Sec. 2 presents key definitions for boosting; Sec. 3 describes our leveraged k-nearest neighbor rule and our UNN learning algorithm, analyzing its properties; Sec. 4 details our experiments; Sec. 5 discusses results and mention future work. In order not to laden the body of the paper, proofsketches of our theorems have been postponed to the appendices.

2. Empirical and surrogate risks

Unless otherwise stated, bold-faced variables like $\boldsymbol{w}$ denote column vectors (components are $w_i,\ i = 1, 2, ...$), calligraphic upper-cases like $\mathcal{S}$ denote sets, small capitals like M denote matrices and their entries are indicated by double indices like m_{ij}; blackboard faces like $\mathbb{X}$ denote subsets of $\mathbb{R}$, the set of real numbers. We let $\mathcal{X}$ denote a *domain* ($\mathbb{R}^n$, $[0,1]^n$, etc., where n is the number of description variables), whose elements are *observations*. An *example* is an ordered pair $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{X} \times \{-1, +1\}^C$. For any $c = 1, 2, ..., C$, y_{ic} denotes the membership of the example to class (or category) c: it is $+1$ (resp. -1) iff the example belongs to c (resp. does not belong to c). In the most general *multilabel* setting, an example may belong to more than one class.

We suppose given a set of m examples:

$$\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m\} \ . \tag{1}$$

In the most basic framework of supervised classification, one wishes to train a *classifier* on $\mathcal{S}$, *i.e.* build a function $\boldsymbol{h} : \mathcal{X} \to \mathbb{R}^C$ with the objective to minimize its *empirical risk* on $\mathcal{S}$, defined as:

$$\varepsilon^{0/1}(\boldsymbol{h}, \mathcal{S}) \doteq \frac{1}{mC} \sum_{c=1}^{C} \sum_{i=1}^{m} 1_{[\varrho(\boldsymbol{h},i,c)<0]} \ , \tag{2}$$

with $1_{[.]}$ the indicator function, called here *0/1 loss*, and

$$\varrho(\boldsymbol{h}, i, c) \ \doteq \ y_{ic} h_c(\boldsymbol{x}_i) \tag{3}$$

the *edge* of classifier $\boldsymbol{h}$ on example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ for class c (analogous to the classification margin: see [11] for details). The sign of h_c in $\{-1, +1\}$ is taken as its membership

another function (here, the non-convex non-differentiable empirical risk). See [9] for details.

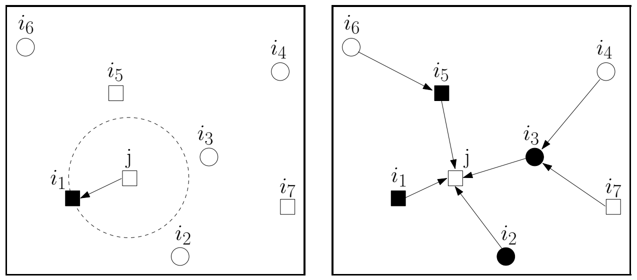


Figure 1: A toy example of *direct* (left) and *reciprocal* (right) k-nearest neighbors ($k = 1$) of an example j. Squares and circles represent examples of positive and negative classes. Each arrow connects an example to its 1-NN.

prediction for class c, thus the edge is positive (resp. negative) when the membership predicted by classifier and the actual example's membership agree (resp. disagree). Therefore, (2) averages over all classes the number of mismatches for the membership predictions. Instead of addressing the empirical risk minimization (2), the current trend of supervised learning, led by boosting and support vector machines, involves minimizing a so-called *surrogate risk* [12]. *E.g.*, boosting relies on summing (or averaging) over classes and examples a so-called real-valued *surrogate loss* ψ. Surrogates are *upper bounds* of the empirical risk with desirable convexity properties. Namely, they provide a convenient *primer* for the maximization of edges, which roughly amounts to finding the "true" predictions ($\varrho(\boldsymbol{h}^\ell, i, c)) > 0$) with large "confidence" values ($|\varrho(\boldsymbol{h}^\ell, i, c))| \gg 0$). Therefore, the minimization of surrogates remarkably impacts on that of the empirical risk, thus enabling to provide optimization algorithms with good generalization properties.

In this paper we follow this approach and constrain ψ to meet the following conditions:

(i) $\psi(z) > 0,\ \forall z \in \mathbb{R}$;

(ii) $\nabla_\psi(0) < 0$ (∇_ψ is the conventional derivative);

(iii) ψ is strictly convex and differentiable.

Conditions **(i)** and **(ii)** imply that ψ is *classification-calibrated*: its local minimization is roughly tied up to that of the empirical risk. **(iii)** implies convenient algorithmic properties for the minimization of the surrogate risk [9]. We end up with the following replacement for (2):

$$\varepsilon^{\psi}(\boldsymbol{h}, \mathcal{S}) \doteq \frac{1}{mC} \sum_{c=1}^{C} \sum_{i=1}^{m} \psi(\varrho(\boldsymbol{h}, i, c)) \ . \tag{4}$$

Some important choices available for ψ include exponential loss, squared loss and logistic loss. Their definitions are displayed in the first column of Table 1.

3. Leveraging k -nearest neighbors

3.1. Leveraged k-NN *for surrogate risk minimization*

We suppose given a non-negative real-valued function, which quantifies to which extent two examples $\boldsymbol{x}$ and $\boldsymbol{x}'$

differ from each other. (For simplicity of notation, we omit index i in $\boldsymbol{x}_i$.) $\mathrm{NN}_k(\boldsymbol{x})$ is defined to be the set of $k > 0$ nearest neighbors for observation $\boldsymbol{x} \in \mathcal{X}$: it contains the indexes of the k examples of $\mathcal{S}$ whose observations are the closest to $\boldsymbol{x}$. (Ties are broken at random.) Similarly, the *reciprocal nearest neighbors* $\mathrm{RNN}_k(\boldsymbol{x'})$ of an observation $\boldsymbol{x'} \in \mathcal{X}$ are those examples in $\mathcal{S}$ for which $\boldsymbol{x'}$ belongs to $\mathrm{NN}_k(.)$. See the illustration in Fig. 1.

The k-nearest neighbors (k-NN) rule is based on a simple majority vote [1], and is defined as the following classifier $\boldsymbol{h}$ (k is implicit):

$$\boldsymbol{h}(\boldsymbol{x}) \doteq \sum_{j \in \mathrm{NN}_k(\boldsymbol{x})} \boldsymbol{y}_j = \sum_{j \in \mathrm{NN}_k(\boldsymbol{x})} \boldsymbol{diag}(\boldsymbol{1}\boldsymbol{y}_j^\top) \ , \tag{5}$$

where $\boldsymbol{1} \in \mathbb{R}^C$ is the all-1 vector, and $\boldsymbol{diag}(\mathrm{M})$ denotes the diagonal vector of a square matrix M. In this paper we propose a *leveraged* k-NN rule $\boldsymbol{h}^\ell$, that consists in the following generalization of (5):

$$\boldsymbol{h}^\ell(\boldsymbol{x}) \doteq \sum_{j \in \mathrm{NN}_k(\boldsymbol{x})} \boldsymbol{diag}(\boldsymbol{\alpha}_j \boldsymbol{y}_j^\top) \ , \tag{6}$$

where $\boldsymbol{\alpha}_j \in \mathbb{R}^C$ is the leveraging coefficient vector for example $(\boldsymbol{x}_j, \boldsymbol{y}_j)$, for $j = 1, ..., m$. We let $\mathrm{R}^{(c)} \in \mathbb{R}^{m \times m}$ ($c = 1, 2, ..., C$) denote the k-NN *edge matrix* for class c:

$$\mathrm{r}_{ij}^{(c)} \doteq \begin{cases} y_{ic} y_{jc} & \text{if} \quad j \in \mathrm{NN}_k(\boldsymbol{x}_i) \\ 0 & \text{otherwise} \end{cases} . \tag{7}$$

The name of $\mathrm{R}^{(c)}$ is justified by an immediate parallel with (3), as each example $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ serves as a classifier for each example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, predicting 0 if $j \notin \mathrm{NN}_k(\boldsymbol{x}_i)$ and y_{jc} otherwise for the membership to class c. Thus $\boldsymbol{r}_j^{(c)}$ (the j^{th} column of $\mathrm{R}^{(c)}$), which we assume different from $\boldsymbol{0}$, collects all edges of "classifier" $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ for class c. It finally comes that the edge of the leveraged k-NN rule on example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ for class c is:

$$\varrho(\boldsymbol{h}^\ell, i, c) = (\mathrm{R}^{(c)} \boldsymbol{\alpha}^{(c)})_i \ , \ c = 1, 2, ..., C \ , \tag{8}$$

where $\boldsymbol{\alpha}^{(c)}$ collects all leveraging coefficients for class c: $\alpha_i^{(c)} \doteq \alpha_{ic}$ ($i = 1, 2, ..., m$). Since choosing $k > 0$ is sufficient for building a meaningful $\mathrm{R}^{(c)}$ ($c = 1, 2, ..., C$), the induction of the leveraged k-NN $\boldsymbol{h}^\ell$ amounts to fitting all $\boldsymbol{\alpha}^{(c)}$s to minimize (4), after replacing the argument of $\psi(\cdot)$ in (4) by (8).

3.2. A boosting algorithm to learn h^ℓ: UNN

We propose a novel classification algorithm, called UNN (Universal Nearest Neighbor rule), whose theoretical basis grounds on recent work of Nielsen and Nock [9] generalizing the boosting approach for the case of linear separators. We tailor our UNN algorithm to the induction of $\boldsymbol{h}^\ell$ (Eq. 6) in the most general multiclass, multilabel classification framework. (Pseudo-code is shown in Alg. 1.) In particular, UNN operates on a set of weights

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

```
Input: $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, ..., m, \ \boldsymbol{x_i} \in \mathcal{X}, \ \boldsymbol{y_i} \in \{-1, +1\}^C\}$, $\psi$ meeting (i), (ii), (iii) (Sec. 2)
$\forall i, j = 1, ..., m, \ c = 1, ..., C$, set:
    $\mathrm{r}_{ij}^{(c)} \doteq \begin{cases} y_{ic} y_{jc} & \text{if} \quad j \in \mathrm{NN}_k(\boldsymbol{x}_i) \\ 0 & \text{otherwise} \end{cases}$      (9)
Let $\boldsymbol{\alpha}_j \leftarrow \boldsymbol{0} \in \mathbb{R}^m$ for $c = 1, 2, ..., C$ do
    Set $\boldsymbol{w} \leftarrow -\nabla_\psi(0)\boldsymbol{1} \in \mathrm{R}_{+*}^m$ for $t = 1$ to $T$ do
        [I.0] Let $j \leftarrow \mathrm{WIC}(\{1, 2, ..., m\}, t, c)$
        [I.1] Let $\delta_j \in \mathbb{R}$ solution of:
            $\sum_{i=1}^m \mathrm{r}_{ij}^{(c)} \nabla_\psi \left(\delta_j \mathrm{r}_{ij}^{(c)} + \nabla_\psi^{-1}(-w_i)\right) = 0$      (10)
        [I.2] $\forall i = 1, 2, ..., m$, set:
            $w_i \leftarrow -\nabla_\psi \left(\delta_j \mathrm{r}_{ij}^{(c)} + \nabla_\psi^{-1}(-w_i)\right)$      (11)
        (Only $w_i$ for which $j \in \mathrm{NN}_k(\boldsymbol{x_i})$ are updated)
        [I.3] Let $\alpha_{jc} \leftarrow \alpha_{jc} + \delta_j$
Output: $\boldsymbol{h}^\ell(\boldsymbol{x}) \doteq \sum_{j \in \mathrm{NN}_k(\boldsymbol{x})} \boldsymbol{diag}(\boldsymbol{\alpha}_j \boldsymbol{y}_j^\top)$
```

$\mathbb{W} \doteq -\mathrm{im}(\nabla_\psi) \subseteq \mathbb{R}$ in surjection with all possible leveraged k-NN rules. (im denotes the "image" set.) The training algorithm repeatedly updates a weight vector $\boldsymbol{w} \in \mathbb{W}$ in order to fit each $\boldsymbol{\alpha}^{(c)}$ ($c = 1, 2, ..., C$). At each iteration t, index $j \in \{1, 2, ..., m\}$ of the example to leverage is obtained by a call to a *weak index chooser* oracle WIC(., ., .), whose implementation is postponed to steps **[I.0.a]** and **[I.0.b]** below. Roughly speaking, learning the leveraging coefficient of an example for a given class amounts to compute its relevance as a prototype of that class, *i.e.*, its significance when voting for an unlabeled observation. Hence, the most relevant prototypes are expected to have large α_{jc} (in absolute value), and vice-versa.

The depiction of UNN is simple, but its implementation has been obviously optimized: *e.g.*, we do not store matrices $\mathrm{R}^{(c)}, c = 1, 2, ..., C$, as (10) only involves indexes i of the reciprocal k-nearest neighbors of example $(\boldsymbol{x}_j, \boldsymbol{y}_j)$, and so on.

The main bottleneck of UNN is step **[I.1]**, as (10) is a non-linear equation. However, it always admits a solution, finite under mild assumptions [9]: namely, δ_j is guaranteed to be finite when there is no total matching or mismatching of example $(\boldsymbol{x}_j, \boldsymbol{y}_j)$'s memberships with its reciprocal neighbors', for the class at hand. The second column of Table 1 contains the solutions to (10) for surrogate losses mentioned in Sec. 2. The following notation is used there:

$$w_j^{(c)+} \doteq \sum_{i: \mathrm{r}_{ij}^{(c)} = 1} w_i, \quad w_j^{(c)-} \doteq \sum_{i: \mathrm{r}_{ij}^{(c)} = -1} w_i \ . \tag{12}$$

Table 1: Three losses and the corresponding solutions δ_j of (10) and w_i of (11). (Vector $\boldsymbol{r}_j^{(c)}$ designates column j of $\mathrm{R}^{(c)}$ and $||.||_1$ the L_1 norm.) The rightmost column says whether it is (A)lways the solution, or whether it is when weights of the reciprocal neighbors $\mathrm{RNN}_k(\boldsymbol{x}_j)$ (Sec. 3.1) are the (S)ame.

loss function	δ_j in (10)	w_i in (11)	Opt
$\psi^{\mathrm{exp}} \doteq \exp\{-x\}$	$\frac{1}{2}\log\frac{w_j^{(c)+}}{w_j^{(c)-}}$	$w_i \exp\left\{-\delta_j \mathrm{r}_{ij}^{(c)}\right\}$	A
$\psi^{\mathrm{squ}} \doteq (1-x)^2$	$\frac{w_j^{(c)+} - w_j^{(c)-}}{2\|\|\boldsymbol{r}_j^{(c)}\|\|_1}$	$w_i - 2\delta_j \mathrm{r}_{ij}^{(c)}$	A
$\psi^{\mathrm{log}} \doteq \log(1+\exp\{-x\})$	$\log\frac{w_j^{(c)+}}{w_j^{(c)-}}$	$\frac{w_i \exp\left\{-\delta_j \mathrm{r}_{ij}^{(c)}\right\}}{1-w_i\left(1+\exp\left\{-\delta_j \mathrm{r}_{ij}^{(c)}\right\}\right)}$	S

Those solutions are always exact for the exponential loss (ψ^{exp}) and squared loss (ψ^{squ}); for the logistic loss (ψ^{log}) it is exact when the weights in the reciprocal neighborhood of $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ are the same, otherwise it is approximated. Since the starting weights are all the same, exactness can be guaranteed during a large number of inner rounds depending on the order in the choice of the examples.

Table 1 also helps to formalize the finiteness condition on δ_j mentioned above: when either sum of weights $w_j^{(c)+}$, $w_j^{(c)-}$ is zero, the solutions in the first and third line are not finite. A simple strategy to cope with numerical problems arising from such situations is that proposed by Schapire and Singer [13]. Namely, in order to compute solutions of δ_j for ψ^{exp} and ψ^{log} (first and third line) we suggest to replace:

$$w_j^{(c)+} \leftarrow w_j^{(c)+} + \frac{1}{m}, \quad w_j^{(c)-} \leftarrow w_j^{(c)-} + \frac{1}{m} \ . \tag{13}$$

Smoothing out δ_j, this guarantees its finiteness without impairing convergence. Indeed, this corresponds to translating the same objective function (4) by a constant regularization offset that guarantees the solution to exist, while still being unique.

Furthermore, Table 1 also shows what the weight update rule (11) specializes to for the mentioned losses.

To finish up, we propose two main alternatives to implement oracle $\textsc{Wic}(.,.,.)$ in step **[I.0]**:

[I.0.a] a lazy approach: we let $\textsc{Wic}(\{1, 2, ..., m\}, t, c) \doteq t$, setting $T = m$;

[I.0.b] the boosting approach: we pick $T \leq m$, and let $\textsc{Wic}$ choose some j for which δ_j is large enough. Each j can be chosen more than once.

Schemes *mixing* **[I.0.a]** and **[I.0.b]** may also be considered: for example, we may pick $T = m$, choose j as in **[I.0.b]**, but exactly once as in **[I.0.a]**. Notice that setting $T < m$ amounts to automatically discarding some examples in $\mathcal{S}$, thus retaining only a subset of annotated data for classification. In the following experiments, we always considered the boosting approach, which generally provides the best precision using less prototypes.

3.3. Properties of UNN

In this section we enunciate two fundamental theorems for UNN. The first one is very general, as it states that the time spent in the induction buys the optimal decrease of the surrogate risk at hand (whatever it is) as T increases. The second theorem refers to the exponential loss minimization, by providing an effective convergence bound. (Proofs for these theorems are given in the appendices.)

Theorem 3.1. *As T increases,* UNN *converges to* $\boldsymbol{h}^\ell$ *realizing the **global** minimum of the surrogate risk at hand (4), for any ψ meeting conditions **(i)**, **(ii)** and **(iii)** above.*

This is the first result of this type known for k-nearest neighbor rules. Although we have proved the boosting ability of UNN for all applicable surrogate losses, we choose to show its behaviour only for the exponential loss ψ^{exp}, which features far better convergence bound than the general one [9].

Computing this bound is based on defining a *weak index assumption* (**WIA**), which is to nearest neighbors what the conventional *weak learning assumption* is to general induced classifiers [13]:

(WIA) let $p_j^{(c)} \doteq w_j^{(c)+}/(w_j^{(c)+} + w_j^{(c)-})$. There exist some $\gamma > 0$ and $\eta > 0$ such that the following holds for index j returned by $\textsc{Wic}(.,.,.)$:

$$|p_j^{(c)} - 1/2| \quad \geq \quad \gamma \ , \tag{14}$$

$$(w_j^{(c)+} + w_j^{(c)-})/||\boldsymbol{w}||_1 \quad \geq \quad \eta \ . \tag{15}$$

Theorem 3.2. *If the **WIA** holds for $\tau \leq T$ steps in* UNN *(for each c), then $\varepsilon^{0/1}(\boldsymbol{h}^\ell, \mathcal{S}) \leq \exp(-2\eta\gamma^2\tau)$.*

Inequality (14) is the usual weak learning assumption [13], when considering examples as weak classifiers. But a *weak coverage assumption* (15) is needed as well, because insufficient *coverage* of the reciprocal (or symmetrized) neighbors could easily wipe out even the surrogate risk reduction potentially due to a large γ. In addition, even when classes are significantly overlapping, choosing k not too small is enough for the **WIA** to be met for a large number of boosting rounds τ, thus determining a potential harsh decrease of $\varepsilon^{0/1}(\boldsymbol{h}^{\ell}, \mathcal{S})$. This is important, as there are at most m different weak classifiers available to Wic(.,.,.), even when each one may be chosen more than once under the **WIA**. Last but not least, Theorem 3.2 also displays the fact that classification (14) may be more important than coverage (15).

4. Experiments

In this section we present experimental results of UNN on both synthetic and real datasets. Our experiments aim to carefully quantify and explain the gains brought by boosting on nearest neighbor voting [7]. First, we performed experiments on synthetic data to drill down into the performances of UNN (Sec. 4.1). Then, we carried out experiments on some standard UCI datasets that are commonly employed for testing nearest neighbors-based techniques [14]. We present a comparison on these datasets with both regular k-NN and a state-of-the-art metric learning method (Sec. 4.2). For both synthetic and real data we used UNN with the exponential loss $\psi^{\exp}$ smoothed out with Eq. (13). In the following, we replace parameter T by $\theta = \frac{T}{m}$, *i.e.*, the proportion of training examples that are selected during the training phase in order to form the final UNN classifier. Thus, parameter θ implicitely determines a cut-off value for coefficients α_{jc} when filtering out the least relevant examples. In all the reported experiments, we fixed the value of θ before running our algorithm, then we evaluated the classification performances as a function of it. (Results of UNN on UCI data refer to the best θ determined by cross-validation.)

4.1. Synthetic data

We have drilled down into the experimental behaviour of UNN using the synthetic Ripley's dataset [10] with $C = 2$ classes. Each population of this dataset is an equal mixture of two two-dimensional normally distributed populations, and the two populations are equally likely. Following the classic setting for this dataset, training and test dataset consist of 250 and 1000 points, respectively, whereas the best theoretical error rate of the Bayes rule is 8.0% [10].

Two main conclusions can be drawn from classification experiments shown in Fig. 2, where performances of UNN are compared to those of k-NN with random sampling, for $k = 5$ (k-NN results are averaged over five of runs). First, UNN consistently outperforms the uniform voting for any value of θ, *i.e.*, the proportion of examples retained for classification, thus confirming UNN as a far more effective data selection technique than random sampling. Second, training a sparse subset of annotated examples with UNN does not degrade classification performances, rather significantly improves them. Namely, notice the points at $\theta = 0.6$ and $\theta = 0.7$, where a sharp performance degradation occurs, due to forcing classification via less and less reliable prototypes, which are instead discarded at lower thresholds. In particular, decreasing θ down to values as small as $\theta = 0.1$ reduces the test error, until reaching misclassification rate very close to Bayes'. (Notice the 3% gap between using all the examples and retaining the smallest subset.)

Indeed, assuming standard sampling assumptions [15], filtering actually benefits from two positive effects. The first is a *margin* effect, well known for *induced* classifiers [15]. The goodness-of-fit of the k-NN rule is driven by the most accurate examples, *i.e.* those surrounded by examples of the same class, getting the largest α_{jc}. The least accurate ones, *e.g.* those located in overlapping regions between two classes, get the smallest (see expressions for δ_j in Table 1). Discarding these latter examples tends to increase a gap between class clouds, but each cloud may shelter examples of different classes. Fortunately, filtering with boosting is accompanied by a subtle local *repolarization* of predictions which, as explained in Figure 3(c), makes that this gap maximization translates to *margin maximization*, for which positive effects on learning are known. The second effect is structural: in nearest neighbor rules, the frontier between classes stems from the Voronoi cells of those least accurate examples. Their discarding separates better the classes, as witnessed by Fig. 3(c). Above all, it reduces the number of Voronoi cells involved in the class frontiers, thus reducing structural parameters (VC-dimension) of the classifier, possibly buying a reduction of the test error as well.

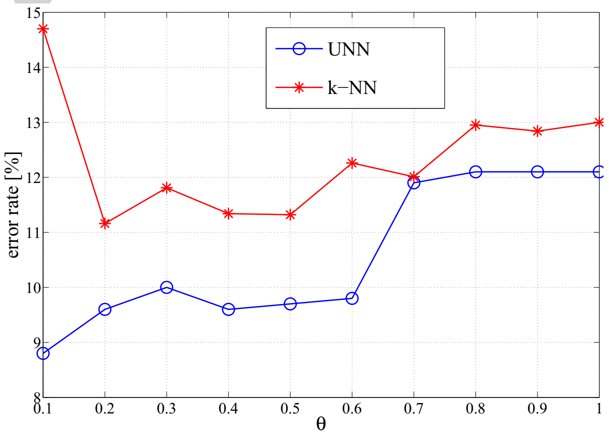
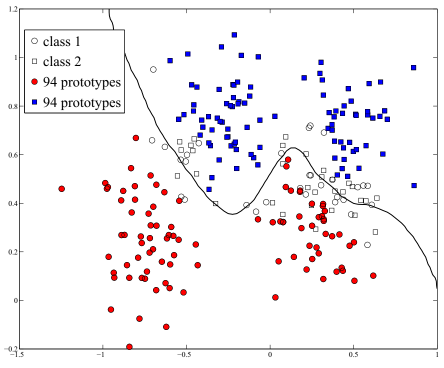


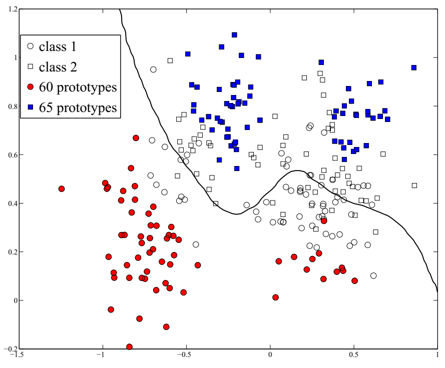
Figure 2: Test error for UNN and k-NN (random sampling) on the Ripley's dataset as a function of the proportion θ of examples retained for classification. Bayes rule (the optimal classifier) achieves 8% error.

4.2. UCI datasets

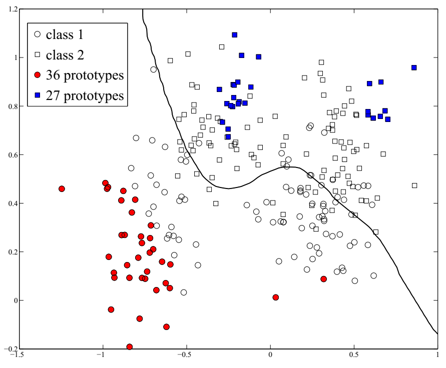
We carried out classification experiments with UNN on standard UCI datasets, and compared our method with both regular k-NN classification (with respect to the Euclidean distance) and ITML [3], that is a state-of-the-art metric learning algorithm. Both UNN and ITML aim at improving the generalization ability of the k-NN rule, but they address this issue from two complementary points of



(a)



(b)



(c)

Figure 3: Maps of Ripley's training data and learned prototypes for $k = 3$ and (a) $\theta = 0.75$, (b) $\theta = 0.5$, (c) $\theta = 0.25$. Examples of class 1 (filled circles) and those of class 2 (filled squares) with the largest α_{jc} are retained as prototypes, thus providing the shown boundary.

view. On the one hand, ITML aims at selecting a suitable distance measure for k-NN search, thus not modifying the uniform voting scheme among neighbor examples. In particular, ITML learns a "Mahalanobis distance" in an information-theoretic framework, where pairwise constraints between training data are incorporated. Then, the labels of nearest neighbors are combined following the regular majority voting rule. On the other hand, our UNN algorithm aims at selecting the most reliable instances to be combined in a weighted voting scheme, without the need to modify the underlying distance measure in the feature space. As a result, the time complexity of UNN is generally much lower than that of metric learning-based approaches, both in training and classification stage. Results of our experiments show that, in most cases, rejecting the less reliable instances from the training dataset, as carried out by UNN, outperforms expensive optimizations of the distance metric provided by ITML.

Results of ITML were produced using the Matlab code provided by the authors [3], using the same settings as in their paper. We evaluated classification performances on five runs of two-fold cross-validation. In Table 2 we present average results over all cross-validation runs, as well as the values used for k. (We found the optimal k value on each dataset by cross-validation.) The same results are presented as histograms in Fig. 4, where the Binomial confidence intervals at the 95% level are shown as well. Globally, UNN outperforms both ITML and k-NN. In particular, UNN outperforms regular k-NN classification (except for the *"glass"* dataset), with the additional advantage of considerably reducing the time complexity of the classification phase, thanks to data filtering. Furthermore, the accuracy improvement of our method over ITML is significant (according to the confidence intervals computed) on some medical datasets, like *"liver"* (6% improvement), *"diabetes"* (3%) and *"cancer"* (1%). Indeed, such data are expected to be more subject to "noisy" examples, due to the high inter-patient variability of medical measurements. All reported results refer to choosing the value of θ by cross-validation. (In most cases, no more than 40% of the training data were retained.)

Table 2: Classification accuracies for UNN, ITML and regular k-NN on various UCI data sets. For each dataset, the best performing method is highlighted by bold digits.

Dataset	k	UNN	ITML	k-NN
Iris	4	**3.07**	3.47	5.73
Balance Scale	4	12.77	**11.46**	19.71
Ionosphere	4	**12.36**	12.65	14.07
Glass	1	3.83	3.18	**2.52**
Liver	8	**32.41**	38.49	33.62
Cancer	6	**6.15**	7.21	7.84
Diabetes	5	**25.44**	28.78	28.10

5. Discussion and conclusion

In this paper, we contribute to fill an important void of nearest neighbor methods [7], showing the way boosting can be transfered to k-NN classification. Experiments on both synthetic and real datasets display that our UNN algorithm provides significant performance improvements not only over regular k-NN, but also over a state-of-the-art metric learning technique (ITML). Furthermore, UNN allows consistent data reduction, which results in significant speed-up while improving the classification accuracy. Finally, our ongoing work focuses on incorporating metric learning into UNN, which is expected to further improve performances.

Appendix A. Proofsketch of Theorem 3.1

We show that UNN converges to the global optimum of any surrogate risk (Sec. 2). So, let us consider the surrogate risk (4) for any class $c = 1, 2, ..., C$ (inner sum). $\boldsymbol{w}_t$ denotes the t^{th} weight vector inside the "**for** c" loop

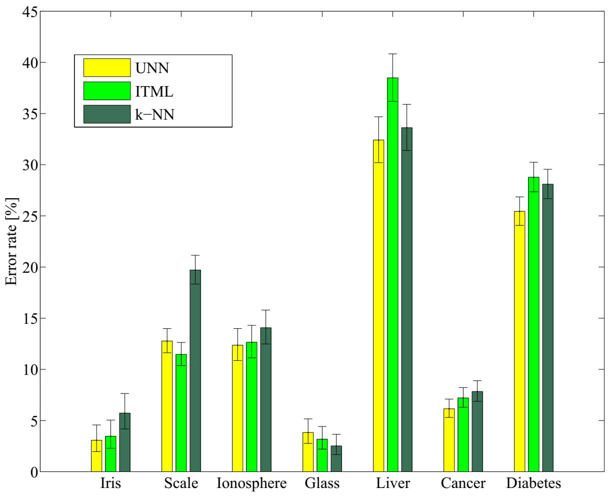


Figure 4: Classification error rates for UNN, ITML and regular k-NN classification. The 95% confidence intervals are also shown.

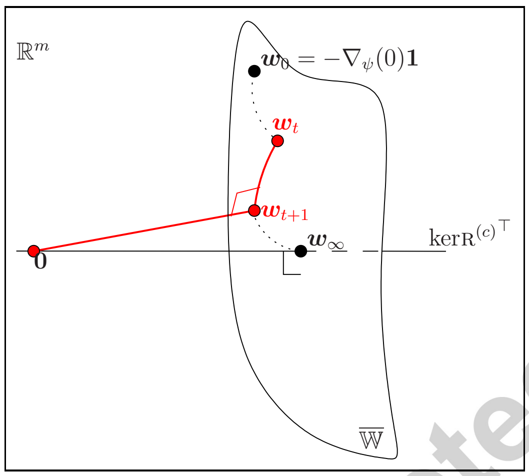


Figure A.5: A geometric view of how UNN converges to the global optimum of (4) (see Appendix for details and notations).

(assuming $\boldsymbol{w}_0$ is the initialization of $\boldsymbol{w}$); similarly, $\boldsymbol{h}_t^\ell$ denotes the t^{th} leveraged k-NN rule obtained after the update in **[I.3]**. The following identity holds, whose prove follows from [9]:

$$\psi(\varrho(\boldsymbol{h}_t^\ell, i, c)) \quad = \quad g + D_{\tilde{\psi}}\left(0 \| w_{ti}\right) \quad , \qquad \text{(A.1)}$$

where $g(m) \doteq -\tilde{\psi}(0)$ does not depend on the k-NN rule. (A.1) makes the connection between the real-valued *classification* problem and a *geometric* problem in the non-metric space of weights. Here, we have made use of the following notations:

- $\tilde{\psi}(x) \doteq \psi^\star(-x)$, where $\psi^\star(x) \doteq x\nabla_\psi^{-1}(x) - \psi(\nabla_\psi^{-1}(x))$ is the Legendre conjugate of ψ, also strictly convex and differentiable.
- $D_{\tilde{\psi}}(w_i \| w'_i) \doteq \tilde{\psi}(w_i) - \tilde{\psi}(w'_i) - (w_i - w'_i)\nabla_{\tilde{\psi}}(w'_i)$ is the Bregman divergence with generator $\tilde{\psi}$. $\psi^\star$ is related to ψ in such a way that $\nabla_{\tilde{\psi}}(x) = -\nabla_\psi^{-1}(-x)$.

(A.1) proves in handy as one computes the *difference* $\varepsilon_c^\psi(\boldsymbol{h}_{t+1}^\ell, \mathcal{S}) - \varepsilon_c^\psi(\boldsymbol{h}_t^\ell, \mathcal{S})$. Indeed, using (A.1) in (4), and computing δ_j in (10) so as to bring $\boldsymbol{h}_{t+1}^\ell$ from $\boldsymbol{h}_t^\ell$, we obtain:

$$\varepsilon_c^\psi(\boldsymbol{h}_{t+1}^\ell, \mathcal{S}) - \varepsilon_c^\psi(\boldsymbol{h}_t^\ell, \mathcal{S}) = -\frac{1}{m}\sum_{i=1}^m D_{\tilde{\psi}}\left(w_{(t+1)i} \| w_{ti}\right). \qquad \text{(A.2)}$$

Since Bregman divergences are non negative and meet the identity of the indiscernibles, (A.2) implies that steps **[I.1]** — **[I.3]** *guarantee* the decrease of (4) as long as $\delta_j \neq 0$. But (4) is lowerbounded, hence UNN must converge. In addition, it converges to the global optimum of (4). Since predictions for each class are independent, the prove consists in showing that the inner sum in (4) converges to its global minimum for each c. Assume this convergence for the current class, c. Then, following [9], (10) and (11) imply that, when any possible $\delta_j = 0$, the weight vector, say $\boldsymbol{w}_\infty$, satisfies $\mathrm{R}^{(c)\top}\boldsymbol{w}^\top = \mathbf{0}$, *i.e.*, $\boldsymbol{w}_\infty \in \ker \mathrm{R}^{(c)\top}$, and $\boldsymbol{w}_\infty$ is unique. But the kernel of $\mathrm{R}^{(c)\top}$ and $\overline{\mathbb{W}}$, the closure of $\mathbb{W}$, are provably Bregman orthogonal, thus yielding:

$$\begin{aligned}\underbrace{\sum_{i=1}^m D_{\tilde{\psi}}\left(0 \| w_i\right)}_{m\varepsilon_c^\psi(\boldsymbol{h}^\ell,\mathcal{S})-mg} &= \underbrace{\sum_{i=1}^m D_{\tilde{\psi}}\left(0 \| w_{\infty i}\right)}_{m\varepsilon_c^\psi(\boldsymbol{h}_\infty^\ell,\mathcal{S})-mg} + \\ &+ \underbrace{\sum_{i=1}^m D_{\tilde{\psi}}\left(w_{\infty i} \| w_i\right)}_{\geq 0}, \forall \boldsymbol{w} \in \overline{\mathbb{W}} \ . \qquad \text{(A.3)}\end{aligned}$$

Underbraces use (A.1) in (4), and $\boldsymbol{h}^\ell$ is a leveraged k-NN rule corresponding to $\boldsymbol{w}$. One obtains that $\boldsymbol{h}_\infty^\ell$ achieves the global minimum of (4), as claimed.

The proofsketch is graphically summarized in Fig. A.5. In particular, two crucial *Bregman orthogonalities* are mentioned. The red one symbolizes:

$$\begin{aligned}\sum_{i=1}^m D_{\tilde{\psi}}\left(0 \| w_{ti}\right) &= \sum_{i=1}^m D_{\tilde{\psi}}\left(0 \| w_{(t+1)i}\right) + \\ &+ \sum_{i=1}^m D_{\tilde{\psi}}\left(w_{(t+1)i} \| w_{ti}\right) \ , \qquad \text{(A.4)}\end{aligned}$$

which is equivalent to (A.2). The black one on $\boldsymbol{w}_\infty$ is (A.3).

Appendix B. Proofsketch of Theorem 3.2

Using developments in [9], UNN can be shown to be equivalent to AdaBoost in which m weak classifiers are available, each one being an example. Each weak classifier returns a value in $\{-1, 0, 1\}$, where 0 is reserved for examples outside the reciprocal neighborhood. Theorem 3 of [13] brings in our case:

$$\varepsilon^{0/1}(\boldsymbol{h}^\ell, \mathcal{S}) \leq \frac{1}{C}\sum_{c=1}^{C}\prod_{t=1}^{T} Z_t^{(c)} \ , \tag{B.1}$$

where $Z_t^{(c)} \doteq \sum_{i=1}^m \tilde{w}_{it}^{(c)}$ is the normalizing coefficient for each weight vector in UNN. ($\tilde{w}_{it}^{(c)}$ denotes the weight of example i at iteration (t, c) of UNN, and the Tilda notation refers to weights normalized to unity at each step.) It follows that:

$$\begin{aligned} Z_t^{(c)} &= 1 - \tilde{w}_{jt}^{(c)+-}\left(1 - 2\sqrt{p_{jt}^{(c)}(1-p_{jt}^{(c)})}\right) \\ &\leq \exp\left(-\tilde{w}_{jt}^{(c)+-}\left(1 - 2\sqrt{p_{jt}^{(c)}(1-p_{jt}^{(c)})}\right)\right) \\ &\leq \exp\left(-\eta\left(1-\sqrt{1-4\gamma^2}\right)\right) \leq \exp(-2\eta\gamma^2) \ , \end{aligned} \tag{B.2}$$

where:

$$\begin{aligned} \tilde{w}_{jt}^{(c)+-} &\doteq \tilde{w}_{jt}^{(c)+} + \tilde{w}_{jt}^{(c)-} \\ p_{jt}^{(c)} &\doteq \tilde{w}_{jt}^{(c)+}/\tilde{w}_{jt}^{(c)+-} = w_{jt}^{(c)+}/w_{jt}^{(c)+-} \ . \end{aligned}$$

The first inequality in (B.2) uses $1 - x \leq \exp(-x)$, and the second the **WIA**. Since even when the **WIA** does not hold, we still observe $Z_t^{(c)} \leq 1$, plugging the last inequality in (B.1) yields the statement of the Theorem.

[1] G. Shakhnarovich, T. Darell, P. Indyk, Nearest-Neighbors Methods in Learning and Vision, MIT Press, 2006.

[2] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1967) 21–27.

[3] J. Davis, B. Kulis, P. Jain, S. Sra, I. Dhillon, Information-theoretic metric learning, in: Proc. of International Conference on Machine Learning (ICML), 2007.

[4] P. Hart, The Condensed Nearest Neighbor rule, IEEE Transactions on Information Theory 14 (1968) 515–516.

[5] H. Brighton, C. Mellish, Advances in instance selection for instance-based learning algorithms, Data Mining and Knowledge Discovery 6 (2002) 153–172.

[6] C. Holmes, N. Adams, Likelihood inference in nearest-neighbour classification models, Biometrika 90 (2003) 99–112.

[7] J.-M. Marin, C.-P. Robert, D.-M. Titterington, A Bayesian reassessment of nearest-neighbor classification, Journal of the Americal Statistical Association 104 (2009) 263–273.

[8] J. R. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81–106.

[9] R. Nock, F. Nielsen, Bregman divergences and surrogates for learning, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2009) 2048–2059.

[10] B. D. Ripley, Neural networks and related methods for classification, Journal of the Royal Statistical Society: Series B 56 (1994) 409–456.

[11] M. Warmuth, J. Liao, G. Rätsch, Totally corrective boosting algorithms that maximize the margin, in: Proc. of International Conference on Machine Learning (ICML), 2006, pp. 1001–1008.

[12] P. Bartlett, M. Jordan, J. D. McAuliffe, Convexity, classification, and risk bounds, Journal of the Americal Statistical Association 101 (2006) 138–156.

[13] R. Schapire, Y. Singer, Improved boosting algorithms using confidence-rated predictions, Machine Learning Journal 37 (1999) 297–336.

[14] A. Asuncion, D. Newman, UCI repository (2007).

[15] R. Schapire, Y. Freund, P. Bartlett, W. Lee, Boosting the margin: a new explanation for the effectiveness of voting methods, Annals of Statistics 26 (1998) 1651–1686.

Paolo Piro defended his PhD thesis in Computer Vision in 2011, January, at the University of Nice-Sophia Antipolis. His research is focused on prototype-based learning methods for classification of real-world and medical images. He is currently Postdoc researcher at the Italian Institute of Technology (IIT), where he is involved in machine learning research for bioinformatics and biomedical image classification.

Richard Nock received the agronomical engineering degree from Agro Montpellier, France, in 1993, the MSc degree in computer science in 1993 the PhD degree in computer science in 1998, and an accreditation to lead research in computer science (HDR) in 2002, all from the University of Montpellier II, France. He joined the Universite Antilles-Guyane in 1998, where he is currently a full professor of computer science. His research interests include machine learning, data mining, computational complexity, and image processing. He received the Best Paper Award at the European Conference on Artificial Intelligence in 2006. His research is being funded by ANR (Young Researcher, White and Thematic programs).

Frank Nielsen prepared his PhD on adaptive computational geometry at INRIA Sophia-Antipolis (France) and defended it in 1996. As a civil servant of the University of Nice (France), he gave lectures at the engineering schools ESSI and ISIA (Ecole des Mines). In 1997, he served in the army as a scientific member in the computer science laboratory of Ecole Polytechnique. In 1998, he joined Sony Computer Science Laboratories Inc., Tokyo (Japan) where he is senior researcher. He became a professor of the CS Dept. of Ecole Polytechnique in 2008. His current research interests focus on computational information geometry. He is a senior ACM and senior IEEE member.

Michel Barlaud is Professor at the University of Nice-Sophia Antipolis. His research deals with image processing: wavelet transform for image and video compression, variationnal methods (PDE's) for image reconstruction, denoising, segmentation, and Statistical methods (Boosting) for classification and content based image retrieval He is editor is chief of two books. He is the co-author of around 250 papers, including 7 book chapters, 60 papers in international journals. He is involved in image processing collaborative research at a national level (ANR/ RNRT networks), at a European level with networks of excellence and International level with collaborations NSF(USA)-CNRS with Stanford University and Boston University. He supervised 30 PhD theses and he was a member of more than 200 PhD committees and he was referee for 6 tenure track promotions in the USA. He served as Associate Editor of the journal IEEE Trans. on Imge Processing. He has been promoted Fellow IEEE for "contributions to the theory and practice of image coding, restoration and segmentation". He has been promoted senior member of Institut Universitaire de France.

***Photo of the author(s)**