# Chapter 12
# Boosting *k*-Nearest Neighbors Classification

**Paolo Piro, Richard Nock, Wafa Bel Haj Ali, Frank Nielsen, and Michel Barlaud**

**Abstract** A major drawback of the *k*-nearest neighbors (*k*-NN) rule is the high variance when dealing with sparse prototype datasets in high dimensions. Most techniques proposed for improving *k*-NN classification rely either on deforming the *k*-NN relationship by learning a distance function or modifying the input space by means of subspace selection. Here we propose a novel boosting approach for generalizing the *k*-NN rule. Namely, we redefine the voting rule as a strong classifier that linearly combines predictions from the *k* closest prototypes. Our algorithm, called UNN (Universal Nearest Neighbors), rely on the *k*-nearest neighbors examples as weak classifiers and learn their weights so as to minimize a *surrogate risk*. These weights, called *leveraging coefficients*, allow us to distinguish the most relevant prototypes for a given class. Results obtained on several scene categorization datasets display the ability of UNN to compete with or beat state-of-the-art methods, while achieving comparatively small training and testing times.

P. Piro (✉)
Istituto Italiano di Tecnologia, Via Morego 30, 16163 Genova, Italy
e-mail: paolo.piro@iit.it

R. Nock
CEREGMIA, Université Antilles-Guyane, Campus de Schoelcher, Martinique, France
e-mail: rnock@martinique.univ-ag.fr

W. Bel Haj Ali · M. Barlaud
I3S Laboratory, University of Nice-Sophia Antipolis, 06903 Sophia Antipolis, France

W. Bel Haj Ali
e-mail: belhajal@i3s.unice.fr

M. Barlaud
e-mail: barlaud@i3s.unice.fr

F. Nielsen
Department of Fundamental Research, Sony Computer Science Laboratories, Inc., Tokyo, Japan

F. Nielsen
LIX Department, Ecole Polytechnique, Palaiseau, France
e-mail: nielsen@lix.polytechnique.fr

G.M. Farinella et al. (eds.), *Advanced Topics in Computer Vision*,
Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-1-4471-5520-1_12, © Springer-Verlag London 2013

## 12.1 Introduction

In this chapter, we describe the proposed approach to $k$-NN boosting. First, we introduce the scope of our work, which aims at automatic visual categorization of scenes (Sect. 12.1.1) and relies on prototype-based classification (Sect. 12.1.2). Then, in Sects. 12.2.1–12.2.3 we present the key definitions for surrogate risk minimization. Our UNN algorithm is detailed in Sect. 12.2.4 for the case of the exponential risk. Section 12.2.5 presents the generic convergence theorem of UNN and the upper bound performance for the exponential risk minimization. Then, in Sect. 12.3, we report our experiments on simulated and real data, comparing UNN with $k$-NN, support vector machines (SVM) and AdaBoost, using Gist and/or Bag-of-Feature descriptors. Real datasets include those proposed in [10, 30, 43], with a number of categories ranging from 8 to 60. Then, in Sects. 12.4 and 12.5 we discuss results, mention future works, and conclude. Finally, we postpone the general form and analysis of UNN to other surrogate risks to the Appendix.

### *12.1.1 Visual Categorization*

In this work, we address the problem of generic visual categorization. This is a relevant task in computer vision, which aims at automatically classifying images into a discrete set of categories, such as *indoor* vs *outdoor* [15, 32], *beaches* vs *mountains*, *churches* vs *towers*. Generic categorization is distinct from object and scene recognition, which are classification tasks concerning particular instances of objects or scenes (e.g., *Notre Dame Cathedral* vs *St. Peter's Basilic*). It is also distinct from other related computer vision tasks, such as content-based image retrieval (that aims at finding images from a database, which are semantically related or visually similar to a given query image) and object detection (which requires to find both the presence and the position of a target object in an image, e.g., person detection).

Automatic categorization of generic scenes is still a challenging task, due to the huge number of natural categories that should be considered in general. In addition, natural image categories may exhibit high inter-class variability (i.e., visually different images may belong to the same category) and low inter-class variability (i.e., distinct categories may contain visually similar images). Classifying images requires an effective and reliable description of the image content, for example, location and shape of specific objects or overall scene appearance. Although several approaches have been proposed in the recent literature to extract semantic information from images [36, 42], most of the state-of-the-art techniques for image categorization still rely on low-level visual information extracted by means of image analysis operators and coded into vector descriptors.

Examples of suitable low-level image descriptors for categorization purposes are Gist, that is, global image features representing the overall scene [30], and SIFT descriptors, that is, descriptors of local features extracted either at salient patches [24] or at dense grid points [23]. A Gist descriptor is based on the so-called "spatial

envelope" [30], which is a very effective low dimensional representation of the overall scene based on spectral information. Such a representation bypasses segmentation, extraction of key-points and processing of individual objects and regions, thus enabling a compact global description of images. Gist descriptors have been successfully used for categorizing locations and environments, showing their ability to provide relevant priors for more specific tasks, like object recognition and detection [40]. Another successful tool for describing the global content of a scene is the Bag-of-Features scheme [38], which represents an image by the histogram of occurrences of vector quantized local descriptors like SIFT.

### *12.1.2 k-NN Classification*

Apart from the descriptors used to compactly represent images, most image categorization methods rely on supervised learning techniques for exploiting information about known samples when classifying an unlabeled sample. Among these techniques, $k$-NN classification has proven successful, thanks to its easy implementation and its good generalization properties [37]. A generalization of the $k$-NN rule to the multi-label classification framework has been also proposed recently by [46], whose technique is based on the maximum-a-posteriori principle applied to multi-labeled $k$-NN. A major advantage of the $k$-NN rule is not to require explicit construction of the feature space and be naturally adapted to multi-class problems. Moreover, from the theoretical point of view, straightforward bounds are known for the true risk (i.e., error) of $k$-NN classification with respect to the Bayes optimum, even for finite samples [29].

Although such advantages make $k$-NN classification very attractive to practitioners, it is an algorithmic challenge to speed-up $k$-NN queries. It is also a statistical challenge to further improve the risk bounds of $k$-NN. In part due to the simplicity of the classification rule, many methods have been proposed to address either of these challenges. For example, many methods have been proposed for speeding up nearest neighbor retrieval, including locality sensitive hashing (LSH, [13]), product quantization for nearest neighbor search [21], and vector space embedding with boosting algorithms [2, 25].

It is yet another challenge to reduce the true risk of the $k$-NN rule, usually tackled by data reduction techniques [17]. In prior work, the classification problem has been reduced to tracking ill-defined categories of neighbors, interpreted as "noisy" [6]. Most of these recent techniques are in fact partial solutions to a larger problem related to the nearest neighbors' true risk, which does not have to be the discrete prediction of labels, but rather a continuous estimation of class membership probabilities [19]. This problem has been reformulated by [7] as a strong advocacy for the formal transposition of *boosting* to nearest neighbors classification. Such a formalization is challenging as nearest neighbors rules are indeed not *induced*, whereas all formal boosting algorithms induce so-called *strong* classifiers by combining *weak* classifiers—also induced, such as decision trees—[35].

A survey of the literature shows that at least four different categories of approaches have been proposed in order to improve $k$-NN classification:

- learning local or global adaptive distance metric;
- embedding data in the feature space (kernel nearest neighbors);
- distance-weighted and difference-weighted nearest neighbors;
- boosting nearest neighbors.

The earliest approaches to generalizing the $k$-NN classification rule relied on learning an adaptive distance metric from training data (see the seminal works of [11]). An analogous approach was later adopted by [18], who carried out linear discriminant analysis to adaptively deform the distance metric. Recently, [31] has proposed a method for learning a weighted distance, where weights can be either global (i.e., only depending on classes and features) or local (i.e., depending on each individual prototype as well).

Other more recent techniques apply the $k$-NN rule to data embedded in a high-dimensional feature space, following the kernel trick approach of support vector machines. For example, [44] have proposed a straightforward adaptation of the kernel mapping to the nearest neighbors rule, which yields significant improvement in terms of classification accuracy. In the context of vision, a successful technique has been proposed by [47], which involves a "refinement" step at classification time, without relying on explicitly learning the distance metric. This method trains a local support vector machine on nearest neighbors of a given query, thus limiting the most expensive computations to a reduced subset of prototypes.

Another class of $k$-NN methods relies on weighting nearest neighbors votes based on their distances to the query sample [8]. Recently, [49] have proposed a similar weighting approach, where the nearest neighbors are weighted based on their vector difference to the query. Such a difference-weight assignment is defined as a constrained optimization problem of sample reconstruction from its neighborhood. The same authors have proposed a kernel-based non-linear version of this algorithm as well.

Finally, comparatively few work have proposed the use of boosting techniques for $k$-NN classification [1, 2, 12, 25, 33]. [1] use AdaBoost for learning a distance function to be used for $k$-NN search. [12] adopt the boosting approach in a non-conventional way. At each iteration a different $k$-NN classifier is trained over a modified input space. Namely, the authors propose two variants of the method, depending on the way the input space is modified. Their first algorithm is based on optimal subspace selection, that is, at each boosting iteration the most relevant subset of input data is computed. The second algorithm relies on modifying the input space by means of non-linear projections. But neither method is strictly an algorithm for inducing weak classifiers from the $k$-NN rule, thus not directly addressing the problem of boosting $k$-NN classifiers. Moreover, such approaches are computationally expensive, as they rely on a genetic algorithm and a neural network, respectively. [2, 25] map examples in a vector space by using the outputs of (Ada)boosted weak classifiers. It is not known whether these algorithms formally keep (or improve) the boosting properties known for AdaBoost [35]. More recently, [33] have built

upon the works of [27, 28] (see also the survey of the approach in [9]) to provide a provable boosting algorithm for $k$-NN classifiers. Guaranteed convergence speed is obtained for AdaBoost's famed exponential loss, under a weak index assumption which parallels the weak learning assumption of boosting algorithms, making the approach of [33] among the first to provide a provable boosting algorithm for k-NN [7].

We propose in this work a full-fledged solution to the problem of boosting $k$-NN classifiers in the general multi-class setting and for general classes of losses. Namely, we propose the first provable boosting algorithm, called UNN, which *induces* a *leveraged* nearest neighbor rule that generalizes the uniform $k$-NN rule, and whose convergence rate is guaranteed for a wide (i.e., infinite) set of losses, encompassing popular choices such as the logistic loss or the squared loss. The voting rule is redefined as a strong classifier that linearly combines weak classifiers of the $k$-NN rule (i.e., the examples). Therefore, our approach does not need to learn a distance function, as it directly operates on the top of $k$-NN search. At the same time, it does not require an explicit computation of the feature space, thus preserving one of the main advantages of prototype-based methods. Our boosting algorithm is an iterative procedure which learns the weights for examples called *leveraging coefficients*. Then, our class encoding allows to generalize the guarantees on convergence *rates* for an *infinite* number of *surrogate risks.*[1] The generalization is highly desirable, not only for experimental purposes related *for example,* to no-free-lunch Theorems [28]: our generalization encompasses many *classification calibrated* surrogates, functions exhibiting particularly convenient guarantees in the context of classification [4]. Finally, an important characteristic of UNN is that it is naturally able, through the leveraging mechanism, to discriminate the most relevant prototypes for a given class.

## 12.2 Method

### *12.2.1 Preliminary Definitions*

In this work, we address the task of *multi-class*, *single-label* image categorization. Although the multi-label framework is quite well established in literature [5], we only consider the case where each image is constrained to belong to one single category among a set of predefined categories. The number of categories (or classes) may range from a few to hundreds, depending on applications. For example, categorization with 67 indoor categories has been recently studied by [34]. We treat the multi-class problem as multiple binary classification problems as it is customary in machine learning. Hence, for each class $c$, a query image is classified either

[1] A *surrogate* is a function which is a suitable upperbound for another function (here, the non-convex non-differentiable empirical risk).

to $c$ or to $\bar{c}$ (the complement class of $c$, which contains all classes but $c$) with a certain confidence (*classification score*). Then the label with the maximum score is assigned to the query. Images are represented by descriptors related to given local or global features. We refer to an image descriptor as an *observation* $\boldsymbol{x} \in \mathcal{X}$, which is a vector of $n$ features and belongs to a *domain* $\mathcal{X}$ (e.g., $\mathbb{R}^n$ or $[0,1]^n$). A label is associated to each image descriptor according to a predefined set of $C$ classes. Hence, an observation with the corresponding label leads to an *example*, which is the ordered pair $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathbb{R}^C$, where $\boldsymbol{y}$ is termed the *class vector* that specifies the class memberships of $\boldsymbol{x}$. In particular, the sign of $y_c$ gives the membership of example $(\boldsymbol{x}, \boldsymbol{y})$ to class $c$, such that $y_c$ is negative iff the observation does not belong to class $c$, positive otherwise. At the same time, the absolute value of $y_c$ may be interpreted as a relative confidence in the membership. Inspired by the multi-class boosting analysis of [48], we constrain class vectors to be *symmetric*, that is:

$$\sum_{c=1}^{C} y_c = 0. \tag{12.1}$$

Hence, in the single-label framework, the class vector of an observation $\boldsymbol{x}$ belonging to class $\tilde{c}$ is defined as:

$$y_{\tilde{c}} = 1, \qquad y_{c \neq \tilde{c}} = -\frac{1}{C-1}. \tag{12.2}$$

This setting turns out to be necessary when treating multi-class classification as multiple binary classifications, as it balances negative and positive labels of a given example over all classes. In the following, we deal with an input set of $m$ examples (or prototypes) $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, \ldots, m\}$, arising from annotated images, which form the *training set*.

### 12.2.2 Surrogate Risks Minimization

We aim at defining a one-versus-all classifier for each category, which is to be trained over the set of examples. This classifier is expected to correctly classify as many new observations as possible, that is, to predict their true labels. Therefore, we aim at determining a classification rule $\boldsymbol{h}$ from the training set, which is able to minimize the classification error over all possible new observations. Since the underlying class probability densities are generally unknown and difficult to estimate, defining a classifier in the framework of supervised learning can be viewed as fitting a classification rule onto a training set $\mathcal{S}$, with the hope to minimize overfitting as well. In the most basic framework of supervised classification, one wishes to train a *classifier* on $\mathcal{S}$, that is, build a function $\boldsymbol{h} : \mathcal{X} \to \mathbb{R}^C$ with the objective to minimize its *empirical risk* on $\mathcal{S}$, defined as:

$$\varepsilon^{0/1}(\boldsymbol{h}, \mathcal{S}) \doteq \frac{1}{mC} \sum_{c=1}^{C} \sum_{i=1}^{m} \big[\varrho(\boldsymbol{h}, i, c) < 0\big], \tag{12.3}$$

with [.] the indicator function (1 iff true, 0 otherwise), called here the 0/1 *loss*, and:

$$\varrho(\boldsymbol{h}, i, c) \doteq y_{ic} h_c(\boldsymbol{x}_i) \tag{12.4}$$

the *edge* of classifier $\boldsymbol{h}$ on example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ for class $c$. Taking the sign of $h_c$ in $\{-1, +1\}$ as its membership prediction for class $c$, one sees that when the edge is positive (resp. negative), the membership predicted by the classifier and the actual example's membership agree (resp. disagree). Therefore, (12.3) averages over all classes the number of mismatches for the membership predictions, thus measuring the goodness-of-fit of the classification rule on the training dataset. Provided the example dataset has good generalization properties with respect to the unknown distribution of possible observations, minimizing this empirical risk is expected to yield good accuracy when classifying unlabeled observations.

However, minimizing the empirical risk is computationally not tractable as it deals with non-convex optimization. In order to bypass this cumbersome optimization challenge, the current trend of supervised learning (including boosting and support vector machines) has replaced the minimization of the empirical risk (12.3) by that of a so-called *surrogate risk* [4], to make the optimization problem amenable. In boosting, it amounts to sum (or average) over classes and examples a real-valued function called the *surrogate loss*, thus ending up with the following rewriting of (12.3):

$$\varepsilon^{\psi}(\boldsymbol{h}, \mathcal{S}) \doteq \frac{1}{mC} \sum_{c=1}^{C} \sum_{i=1}^{m} \psi\big(\varrho(\boldsymbol{h}, i, c)\big). \tag{12.5}$$

Relevant choices available for $\psi$ include:

$$\psi^{\text{sqr}} \doteq (1 - x)^2, \tag{12.6}$$

$$\psi^{\text{exp}} \doteq \exp(-x), \tag{12.7}$$

$$\psi^{\log} \doteq \log\big(1 + \exp(-x)\big); \tag{12.8}$$

(12.6) is the squared loss [4], (12.7) is the exponential loss [35], and (12.8) is the logistic loss [4]. Such surrogates play a fundamental role in supervised learning. They are upper bounds of the empirical risk with desirable convexity properties. Their minimization remarkably impacts on that of the empirical risk, thus enabling to provide minimization algorithms with good generalization properties [28].

In the following, we move from recent advances in boosting with surrogate risks to redefine the $k$-NN classification rule. Our algorithm, UNN (Universal Nearest Neighbors), is first proposed for the exponential surrogate. We describe in the appendix the general formulation of the algorithm, not restricted to this surrogate. We show that UNN converges to the optimum of many surrogates with guaranteed convergence rates under mild assumptions, and more generally converges to the global optimum of the surrogate risk for an even wider set of surrogates.

### 12.2.3 Leveraging the k-NN Rule

We denote by $\mathrm{NN}_k(\boldsymbol{x})$ the set of the $k$-nearest neighbors (with integer constant $k > 0$) of an example $(\boldsymbol{x}, y)$ in set $\mathcal{S}$ with respect to a non-negative real-valued "distance" function. This function is defined on domain $\mathcal{X}$ and measures how much two observations differ from each other. This dissimilarity function thus may not necessarily satisfy the triangle inequality of metrics. For sake of readability, we let $i \sim_k \boldsymbol{x}$ denote an example $(\boldsymbol{x}_i, y_i)$ that belongs to $\mathrm{NN}_k(\boldsymbol{x})$. This neighborhood relationship is intrinsically asymmetric, that is, $\boldsymbol{x}_i \in \mathrm{NN}_k(\boldsymbol{x})$ does not necessarily imply that $\boldsymbol{x} \in \mathrm{NN}_k(\boldsymbol{x}_i)$. Indeed, a nearest neighbor of $\boldsymbol{x}$ does not necessarily contain $\boldsymbol{x}$ among its own nearest neighbors.

The $k$-nearest neighbors rule ($k$-NN) is the following multi-class classifier $\boldsymbol{h} = \{h_c : c = 1, 2, \ldots, C\}$ ($k$ appears in the summation indices):

$$h_c(\boldsymbol{x}) = \sum_{j \sim_k \boldsymbol{x}} [y_{jc} > 0], \tag{12.9}$$

where $h_c$ is the one-versus-all classifier for class $c$ and square brackets denote the indicator function. Hence, the classic nearest neighbors classification is based on majority vote among the $k$ closest prototypes.

We propose to weight the votes of nearest neighbors by means of real coefficients, thus generalizing (12.9) to the following *leveraged* $k$-NN rule $\boldsymbol{h}^\ell = \{h_c^\ell : c = 1, 2, \ldots, C\}$:

$$h_c^\ell(\boldsymbol{x}) = \sum_{j \sim_k \boldsymbol{x}} \alpha_{jc} y_{jc}, \tag{12.10}$$

where $\alpha_{jc} \in \mathbb{R}$ is the leveraging coefficient for example $j$ in class $c$, with $j = 1, 2, \ldots, m$ and $c = 1, 2, \ldots, C$. Hence, (12.10) linearly combines class labels of the $k$ nearest neighbors (defined in Sect. 12.2.1) with their leveraging coefficients.

Our work is focused on formal boosting algorithms working on top of the $k$-NN methods. These algorithms do not affect the nearest neighbor search when inducing weak classifiers of (12.10). They are thus independent on the way nearest neighbors are computed, unlike most of the approaches mentioned in Sect. 12.1.2, which rely on modifying the neighborhood relationship via metric distance deformations or kernel transformations. This makes our approach fully compatible with any underlying (metric) distance and data structure for $k$-NN search, as well as possible kernel transformations of the input space.

For a given training set $\mathcal{S}$ of $m$ labeled examples, we define the $k$-NN *edge matrix* $\mathrm{R}^{(c)} \in \mathbb{R}^{m \times m}$ for each class $c = 1, 2, \ldots, C$:

$$\mathrm{r}_{ij}^{(c)} \doteq \begin{cases} y_{ic} y_{jc} & \text{if } j \sim_k i \\ 0 & \text{otherwise.} \end{cases} \tag{12.11}$$

The name of $\mathrm{R}^{(c)}$ is justified by an immediate parallel with (12.4). Indeed, each example $j$ serves as a classifier for each example $i$, predicting 0 if $j \notin \mathrm{NN}_k(\boldsymbol{x}_i)$,

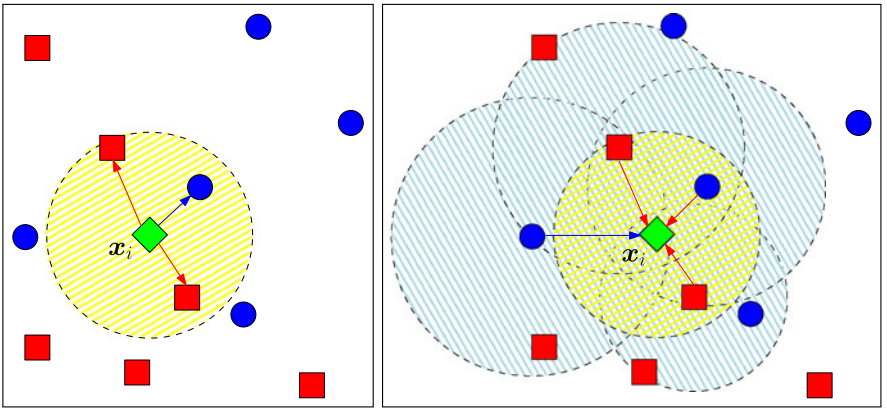


**Fig. 12.1** Schematic illustration of the *direct* (*left*) and *reciprocal* (*right*) $k$-nearest neighbors ($k = 1$) of an example $\boldsymbol{x}_i$ (*green diamond*). *Red squares* and *blue circles* represent examples of positive and negative classes. Each *arrow* connects an example to its $k$-nearest neighbors

$y_{jc}$ otherwise, for the membership to class $c$. Hence, the $j$th column of matrix $\mathrm{R}^{(c)}$, $\boldsymbol{r}_j^{(c)}$, which is different from $\boldsymbol{x}$ when choosing $k > 0$, collects all edges of "classifier" $j$ for class $c$. Note that nonzero entries of this column correspond to the so-called *reciprocal nearest neighbors* of $j$, that is, those examples for which $j$ is a neighbor (Fig. 12.1). Eventually, the edge of the leveraged $k$-NN rule on example $i$ for class $c$ reads:

$$\varrho\big(\boldsymbol{h}^{\ell}, i, c\big) = \big(\mathrm{R}^{(c)}\boldsymbol{\alpha}^{(c)}\big)_i, \quad c = 1, 2, \ldots, C, \tag{12.12}$$

where $\boldsymbol{\alpha}^{(c)}$ collects all leveraging coefficients in a vector form for class $c$: $\alpha_i^{(c)} \doteq \alpha_{ic}$, $i = 1, 2, \ldots, m$. Thus, the induction of the leveraged $k$-NN classifier $\boldsymbol{h}^{\ell}$ amounts to fitting all $\boldsymbol{\alpha}^{(c)}$'s so as to minimize (12.5), after replacing the argument of $\psi(\cdot)$ in (12.5) by (12.12).

### *12.2.4 UNN Boosting Algorithm*

We explain our classification algorithm specialized for the exponential loss minimization in the multi-class one-versus-all framework, with pseudo-code shown in Algorithm 1. Like common boosting algorithms, UNN operates on a set of weights $w_i$ ($i = 1, 2, \ldots, m$) defined over training data. Such weights are repeatedly updated to fit all leveraging coefficients $\boldsymbol{\alpha}^{(c)}$ for class $c$ ($c = 1, 2, \ldots, C$). At each iteration, the index to leverage, $j \in \{1, 2, \ldots, m\}$, is obtained by a call to a *weak index chooser* oracle $\textsc{Wic}(.,.,.)$, whose implementation is detailed later in this section.

Figure 12.2 presents a block diagram of UNN algorithm. In particular, notice how the initialization step, relying on $k$-NN and edge matrix computation, is clearly

**Algorithm 1** Universal Nearest Neighbors UNN($\mathcal{S}$) for $\psi = \psi^{\exp}$

**Input:** $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, \ldots, m,\ \boldsymbol{x}_i \in \mathcal{X},\ \boldsymbol{y}_i \in \{-\frac{1}{C-1}, 1\}^C\}$

$\mathrm{r}_{ij}^{(c)} \doteq \begin{cases} y_{ic} y_{jc} & \text{if } j \sim_k \boldsymbol{x}_i \\ 0 & \text{otherwise} \end{cases} \quad \forall i, j = 1, 2, \ldots, m,\ c = 1, 2, \ldots, C$ ▷ $k$-NN edge matrix

**for** $c = 1, 2, \ldots, C$ **do**

$\alpha_{jc} \leftarrow 0, \quad \forall j = 1, 2, \ldots, m$ ▷ Leveraging coefficients

$w_i \leftarrow 1, \quad \forall i = 1, 2, \ldots, m$ ▷ Example weights

**for** $t = 1, 2, \ldots, T$ **do**

**[I.1]** $j \leftarrow \textsc{Wic}(\{1, 2, \ldots, m\}, t)$ ▷ Weak index chooser oracle

**[I.2]**

$$w_j^+ = \sum_{i : \mathrm{r}_{ij}^{(c)} > 0} w_i, \qquad w_j^- = \sum_{i : \mathrm{r}_{ij}^{(c)} < 0} w_i \tag{12.13}$$

$$\delta_j \leftarrow \frac{1}{2} \log\left(\frac{w_j^+}{w_j^-}\right) \tag{12.14}$$

**[I.3]**

$$w_i \leftarrow w_i \exp\left(-\delta_j \mathrm{r}_{ij}^{(c)}\right), \quad \forall i : j \sim_k \boldsymbol{x}_i \tag{12.15}$$

**[I.4]** $\alpha_{jc} \leftarrow \alpha_{jc} + \delta_j$

**end for**

**end for**

**Output:** $h_c(\boldsymbol{x}) = \sum_{i \sim_k \boldsymbol{x}} \alpha_{ic} y_{ic}, \quad \forall c = 1, 2, \ldots, C$

distinguished from the iterative procedure, where a new prototype is added at each iteration $t$, thus updating both the strong classifier $\boldsymbol{h}(\boldsymbol{x})$ and the weights $w_i$.

The training phase is implemented in a one-versus-all fashion, that is, $C$ learning problems are solved independently, and for each class $c$ the training examples are considered as belonging to either class $c$ or the complement class $\bar{c}$, that is, *any other class*. Eventually, one leveraging coefficient ($\alpha_{jc}$) per class is learned for each weak classifier (indexed by $j$).

The key observation when training weak classifiers with UNN is that, at each iteration, one single example (indexed by $j$) is considered as a prototype to be leveraged. Indeed, all the other training data are to be viewed as observations for which $j$ may possibly vote. In particular, due to $k$-NN voting, $j$ can be a classifier only for its reciprocal nearest neighbors (i.e., those data for which $j$ itself is a neighbor, corresponding to nonzero entries in matrix (12.11) on column $j$). This brings to a remarkable simplification when computing $\delta_j$ in step [I.2] and updating weights $w_i$ in step [I.3] (Eqs. (12.14), (12.15)). Indeed, only weights of reciprocal nearest neighbors of $j$ are involved in these computations, thus allowing us not to store the entire matrix $\mathrm{R}^{(c)}$, $c = 1, 2, \ldots, C$. Note that the set of reciprocal neighbors is split in two subsets, each containing examples that agree (disagree) with the class membership of $j$, thus yielding the partial sums $w_j^+$ and $w_j^-$ of (12.13).

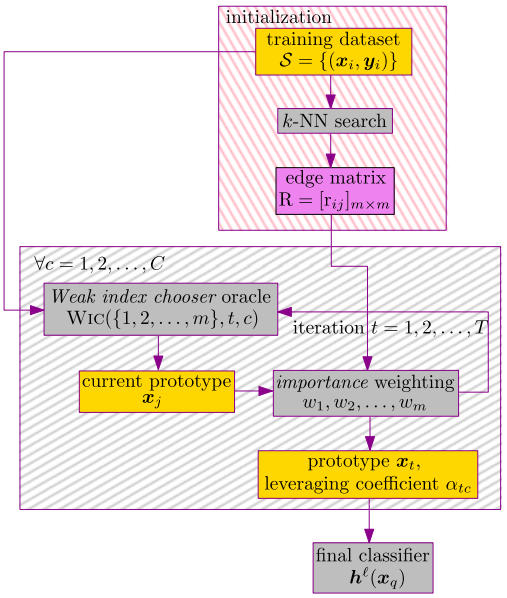


**Fig. 12.2** Block diagram of the UNN learning scheme

Note that when whichever $w_j^+$ or $w_j^-$ is zero, $\delta_j$ in (12.14) is not finite. There is however a simple alternative, inspired by [35], which consists in smoothing out $\delta_j$ when necessary, thus guaranteeing its finiteness without impairing convergence. More precisely, we suggest to replace:

$$w_j^+ \leftarrow w_j^+ + \frac{1}{m}, \tag{12.16}$$

$$w_j^- \leftarrow w_j^- + \frac{1}{m}. \tag{12.17}$$

Also note that step [I.1] relies on oracle $\textsc{Wic}(.,.,.)$ for selecting index $j$ of the next weak classifier. We propose two alternative implementations of this oracle, as follows:

(a) a *lazy* approach: $T = m$, $\textsc{Wic}(\{1, 2, \ldots, m\}, t, c) \doteq t$;
(b) the *boosting* approach: we pick $T \geq m$, and let $j$ be chosen by $\textsc{Wic}(\{1, 2, \ldots, m\}, t, c)$ such that $\delta_j$ is large enough. Each $j$ can be chosen more than once.

There are also schemes *mixing* (a) and (b): for example, we may pick $T = m$, choose $j$ as in (b), but exactly once as in (a).

### 12.2.5 UNN Convergence

The main properties of UNN are summarized by the following three fundamental theorems. The first theorem ensures general *monotonic convergence* to the optimal

surrogate loss, for *any* given surrogate function. The second theorem further refines this general convergence theorem by providing an effective convergence bound for the exponential loss.

Suppose that $\psi$ meets the following conditions:

(i) $\mathrm{im}(\psi) = \mathbb{R}_+$;
(ii) $\nabla_\psi(0) < 0$ ($\nabla_\psi$ is the conventional derivative);
(iii) $\psi$ is strictly convex and differentiable.

**Theorem 12.1** *As the number of iteration steps $T$ increases, UNN converges to $\boldsymbol{h}^\ell$ realizing the* global *minimum of the surrogate risk at hand* (12.5)*, for any $\psi$ meeting conditions* (i), (ii) *and* (iii) *above.*

*Proof* A proofsketch is given in Appendix. □

Then, in order to obtain the specific convergence rate for $\psi^{\exp}$, suppose the following *weak index assumption* (WIA) holds. (See Eq. (12.13) in Algorithm 1 for the definition of $w_j^{(c)+}$ and $w_j^{(c)-}$.)

(WIA) There exist some $\gamma > 0$ and $\eta > 0$ such that the following two inequalities hold for index $j$ returned by $\textsc{Wic}(.,.,.)$:

$$\left| \frac{w_j^{(c)+}}{w_j^{(c)+} + w_j^{(c)-}} - \frac{1}{2} \right| \geq \gamma, \tag{12.18}$$

$$\frac{w_j^{(c)+} + w_j^{(c)-}}{\|\boldsymbol{w}\|_1} \geq \eta. \tag{12.19}$$

**Theorem 12.2** *If the* (WIA) *holds for $v \leq T$ steps in UNN (for each $c$), then* $\varepsilon^{0/1}(\boldsymbol{h}^\ell, \mathcal{S}) \leq \exp(-\Omega(\eta\gamma^2 v))$.

*Proof* A proofsketch is given in Appendix. □

Theorems 12.1 and 12.2 show that UNN converges (exponentially fast) to the global optimum of the surrogate risk on the training set. Most of the recent works that can be associated to boosting algorithms, or more generally to the minimization of some surrogate risk using whichever kind of procedure, have explored the universal consistency of the surrogate minimization problems (see [4, 26, 45], and references therein). The problem can be roughly stated as whether the minimization of the surrogate risk guarantees in probability for the classifier built to converge to the Bayes rule as $m \to \infty$. This question obviously becomes relevant to UNN given our results. Among the results contained in this rich literature, the one whose consequences directly impact on the universal consistency of UNN is Theorem 3 of [4]. We can indeed easily show that all our choices of surrogate loss are classification calibrated, so that minimizing the surrogate risk in the limit ($m \to \infty$) implies minimizing the true risk, and implies uniform consistency as well. Moreover, this result,

proven for $C = 2$, holds as well for arbitrary $C \geq 2$ in the single-label prediction problem. [3] proved an additional result for AdaBoost [35]: if the algorithm is run for a number $T \geq m^{\eta}$ boosting rounds, for $\eta \in (0, 1)$, then there is indeed minimization in the limit of the exponential risk, and so AdaBoost is universally consistent. From our theorems above, this implies the consistency of UNN, and this even has the consequence to prove that the filtering procedure described in the experiments is also consistent, since indeed [3]'s bound implies that we leverage a proportion of $1/m^{1-\eta}$ examples, "filtering out" the remaining ones.

Moreover, the results of [26] are also interesting in our setting, even when they are typically aimed at boosting algorithms with weak learners like decision-tree learning algorithms, that define *quantizations* of the observations (each decision tree defines a new description variable for the examples). They show that there exists conditions on the quantizers that yield conditions on the surrogate loss function for universal consistency. It is interesting to notice that the universal consistency of UNN does not need such assumptions, as weak learners are examples that do not make quantizations of the observation's domain. Finally, the work of [45] explores the consistency of surrogate risk minimization in the case where rejects are allowed by classifiers, somehow refusing to classify an observation at a cost smaller than misclassifying. While this setting is not relevant to UNN in the general case, it becomes relevant as we filter out examples (see the experiments), which boils down to stating that they systematically reject on observations.

On the one hand, [45] show that filtering out examples does not impair UNN universal consistency, as long as filter thresholds are locally based. On the other hand, they also provide a way to quantify the actual loss $\ell_{r,j}$ caused by filtering out example $j$, which we recall is in between 0 (the loss of good classification) and 1 (the loss of bad classification). For example, choosing the exponential loss and using Theorem 1 in [45] reveals that the reject loss is:

$$\ell_{r,j} = \frac{\min\{w_j^+, w_j^-\}}{w_j^+ + w_j^-}.$$

Let us now complete further the picture of boosting algorithms for $k$-NN, by showing that, under a mild additional assumption on $\psi$, we obtain a guaranteed convergence rate for UNN. Of particular interest is the assumption under which we are able to prove this result. Following [27, 28], we make a "Weak Edge Assumption":

(WEA) There exists some $\vartheta > 0$ such that the following inequality holds for index $j$ returned by $\textsc{Wic}(.,.,.)$:

$$\left| \sum_{i : j \sim_k i} \mathrm{r}_{ij}^{(c)} w_i \right| \geq \vartheta. \tag{12.20}$$

This assumption states that the average value (in absolute value) of $y_{ic} y_{jc}$ over the reciprocal neighborhood of example $j$ cannot be smaller than some constant $\vartheta$. It is *weak* for the following reason. If the classes in the reciprocal neighborhood were

picked at random, the quantity inside the absolute value in (12.20) would be zero in average because of the way we model classes in (12.1). So, we are assuming that, regardless of weights, we can always pick an example $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ "beating" random by a potentially small advantage $\vartheta$. Note that (WEA) is weaker than (WIA) in the sense that we do not make any coverage assumption like (12.19).

Let us now turn to the assumption on $\psi$:

(iv) $\psi$ is locally $\omega$ strongly smooth, for some $\omega > 0$:

$$D_\psi\big(x' \| x\big) \leq \frac{\omega}{2}\big(x' - x\big)^2, \tag{12.21}$$

where $x, x'$ range through the values $\varrho(\boldsymbol{h}, i, c)$ over which UNN is run, and

$$D_\psi\big(x' \| x\big) \doteq \psi\big(x'\big) - \psi(x) - \big(x' - x\big)\nabla_\psi(x) \tag{12.22}$$

is the Bregman divergence with generator $\psi$. There is an important duality between strong smoothness and strong convexity, with applications in machine learning and optimization [22]. The proof of the following theorem, in the Appendix, is another example of its applicability in these fields.

**Theorem 12.3** *If the* (WEA) *holds and* $\psi$ *meets assumptions* (i)–(iv), *then for any user-fixed* $\tau \in [0, 1]$, *UNN has fit a leveraged* $k$*-NN classifier with empirical risk no greater than* $\tau$ *provided the number of boosting iterations* $T$ *satisfies*:

$$T \geq \frac{2(1-\tau)\psi(0)\omega k m}{\vartheta^2 (C-1)} = \Omega\left(\frac{\omega k m}{\vartheta^2}\right). \tag{12.23}$$

Theorem 12.3 does not obliterate the (better) convergence results for the exponential loss of Theorem 12.2, yet it opens the guarantees of convergence under weak assumptions to some of the most interesting surrogates in classification. These include *permissible convex surrogates* (PCS, [27]), a set containing as special cases the squared and logistic surrogates in (12.6), (12.8). Informally, any loss which meets regularity conditions and common requirements about losses, such as lower-boundedness, symmetry and the proper scoring property, can be represented by a PCS [27]. The exponential surrogate in (12.7) is not a PCS, yet it is a first-order approximation to the logistic surrogate. Up to translating and scaling by constants, any PCS meets $\mathrm{im}(\nabla_\psi) \subseteq [-1, 0]$ [27]. Reasoning on the second derivative of $\psi$, we see that there is not much room to violate (12.21), thus making many PCS $\omega$ strongly smooth for small values of $\omega$. Simple calculations yield that we can take for example $\omega = 1/4$ for the logistic loss (12.8), and $\omega = 2$ for the squared loss (12.6), making the bound in (12.23) more favorable to the former. As a last example, consider the following parameterized choice for $\psi$, with $\mu \in (0, 1)$:

$$\psi_\mu^{\mathrm{mat}} \doteq \frac{1}{1-\mu}\big(-x + \sqrt{(1-\mu)^2 + x^2}\big);$$

this choice, which gives rise to Matsushita's loss for $\mu = 0$, has important convexity properties [27]. In this case, we easily obtain that we can pick $\omega = 1/(1-\mu)$.

## 12.3 Experiments

In this section, we present experimental results of UNN for image categorization. In order to reduce numerical problems on the large databases on which we test UNN, we normalize weights to unity after the update in (12.15). Our experiments aim at carefully quantifying and explaining the gains brought by boosting on $k$-NN voting on real image databases. In particular, we propose in Sects. 12.3.1 and 12.3.2 an analysis and comparison of UNN vs $k$-NN for Gist and Bag-of-Features descriptors on two broadly used datasets of natural images. In Sect. 12.3.3, we drill down into precision and execution times comparisons between UNN vs $k$-NN, SVM and AdaBoost. We also introduce in this section a soft version of UNN which, to classify new observations, convolutes weighting with a simple density estimation suggested by boosting.

### *12.3.1 Image Categorization Using Global Gist Descriptors*

We tested UNN on global descriptors for the categorization of natural images. In particular, we used the database of natural scenes collected by [30], which has been successfully used to validate several classification techniques relying on Gist image descriptors. A Gist descriptor provides a global representation of a scene directly, without requiring neither an explicit segmentation of image regions and objects nor an intermediate representation by means of local features. In the standard setting, an image is first resized to square, then represented by a single vector of $d$ components (typically $d = 512$ or $d = 320$), which collects features related to the spatial organization of dominant scales and orientations in the image. The one-to-one mapping between images and Gist descriptors is one of the main advantages of using such a global representation instead of local descriptors. In particular, the ability to map any instance to a single point in the feature space is crucial for the effectiveness of $k$-NN methods, where computing the one-to-one similarity between testing and training instances is explicitly required at classification time. Conversely, representing an image with a set of multiple local descriptors is not directly adapted to such discriminative classification techniques, thus generally requiring an intermediate (usually unsupervised) learning step in order to extract a compact single-vector descriptor from the set of local descriptors [14]. For example, this is the case for Bag-of-Features methods, that we discuss in Sect. 12.3.2 along with an experimental comparison to our method. Finally, although Gist is not an alternative image representation method with respect to local descriptors, it has proven very successful in representing relevant contextual information of natural scenes, thus allowing, for instance, to compute meaningful priors for exploration tasks, like object detection and localization [40].

In the following, we denote as 8-*cat* the database of [30], which contains 2,688 color images of outdoor scenes of size $256 \times 256$ pixels, divided in 8 categories: *coast*, *mountain*, *forest*, *open country*, *street*, *inside city*, *tall buildings* and *highways*. One example image of each category is shown in Fig. 12.3. In addition, we


**Fig. 12.3** Examples of annotated images from the 8 categories database of [30]


**Fig. 12.4** Examples of the five additional categories included in the 13 categories database of [10]

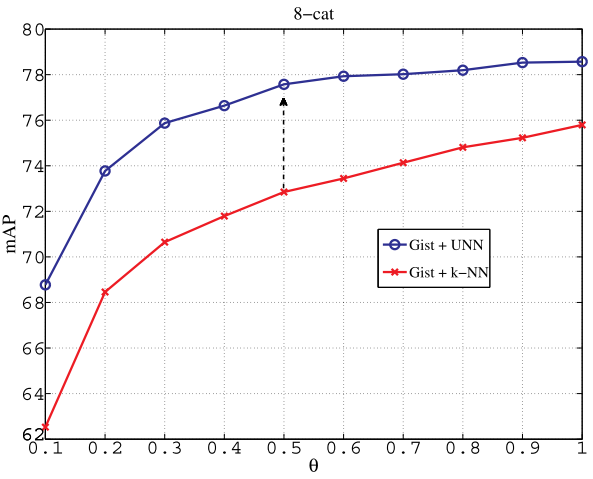


**Fig. 12.5** Gist image classification performances of UNN compared to $k$-NN on the 8-cat database (see text for details)

carried out categorization experiments on a larger database of 13 categories as well, denoted as 13-*cat*. This dataset was firstly proposed by [10] and contains five more categories, as shown in Fig. 12.4. We extracted Gist descriptors from these images with the most common settings: 4 resolution levels of the Gabor pyramid, 8 orientations per scale and $4 \times 4$ blocks.[2]

We evaluated classification performances when filtering the prototype dataset, that is, retaining a proportion $\theta$ of the most relevant examples as prototypes for classification.

[2]The implementation by the authors is available at: http://people.csail.mit.edu/torralba/code/spatialenvelope/sceneRecognition.m.

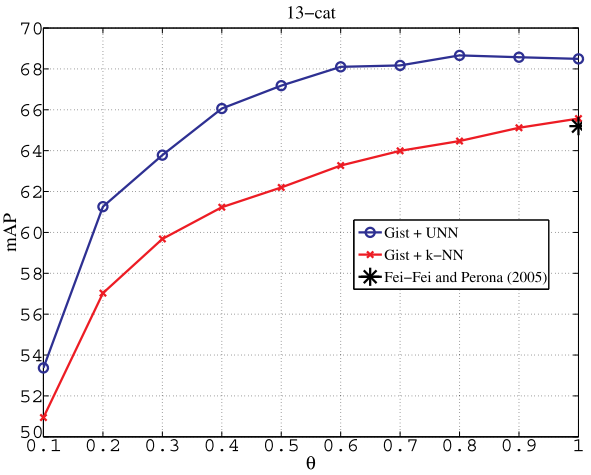


**Fig. 12.6** Gist image classification performances of UNN compared to $k$-NN on the 13-cat database (see text for details)

In Figs. 12.5 and 12.6, we show classification performances in terms of the mean Average Precision (MAP)[3] as a function of $\theta$. We randomly chose half images to form a training set, while testing on the remaining ones. In each UNN experiment, we fixed the value of $\theta = T/m$, thus constraining the number of training iterations $T$ such that at most $T$ examples could be retained as prototypes.

We compared UNN with the classic $k$-NN classification. Namely, in order for the classification cost of $k$-NN be roughly the same as UNN, we carried out random sampling of the prototype dataset for selecting proportion $\theta$ (between 10 % and the whole set of examples). UNN significantly outperforms classic $k$-NN. Take for example $\theta = 0.5$ in Fig. 12.5: UNN not only outperforms $k$-NN with $\theta = 0.5$, its MAP also exceeds that of $k$-NN with all data ($\theta = 1$) by almost 2 %. Moreover, on the 13-cat database, UNN outperforms the technique proposed by [10] by 3 % (the asterisk in Fig. 12.6, which corresponds to the best result reported in their paper).

### *12.3.2 Image Categorization Using Bags-of-Features*

We now describe experiments with UNN on the Bag-of-Features (BoF) image classification approach. This technique is based on extracting a "bag" of local descriptors (e.g., SIFT descriptors) from an image and vector quantizing them on a precomputed vocabulary of so-called "visual words" [38]. An image is then represented by the histogram of visual word frequencies. This approach provides an effective tool

[3]The MAP was computed by averaging classification rates over categories (diagonal of the confusion matrix) and then averaging those values after repeating each experiment 10 times on different folds.

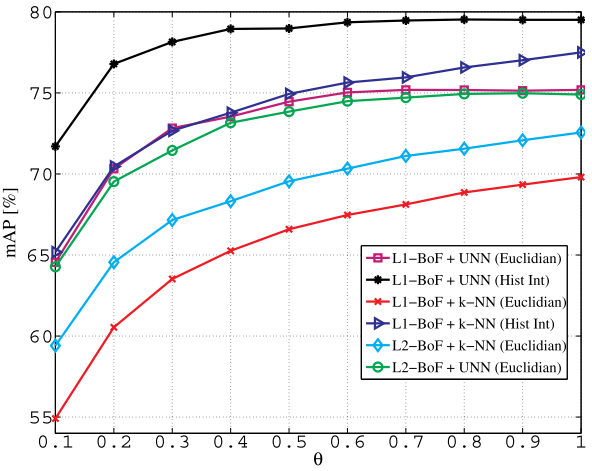


**Fig. 12.7** Overall results of BoF classification with UNN compared to $k$-NN for different settings of histogram normalization (either L1- or L2-norm) and nearest neighbor matching (either Euclidean distance or Histogram Intersection)

for image categorization, as it relies on one single compact descriptor per image, while keeping the informative power of local features. We compare UNN and $k$-NN on the 8-cat database (see Sect. 12.3.1).

We used the VLFeat toolbox [41][4] for extracting gray-scale dense SIFT descriptors at four resolution levels. In particular, a regular grid with spacing 10 pixels was defined over the image and at each grid point SIFT descriptors were computed over circular support patches with radii 4, 8, 12 and 16 pixels. As a result, each point was represented by four different SIFT descriptors. Therefore, given the image size $256 \times 256$, we obtained about 2,500 SIFT descriptors per image. Then we split the database in two distinct subsets of images, half for training and half for testing (i.e., 1,344 images in each dataset). In order to build the dictionary of visual words, we applied $k$-means clustering on 600,000 SIFT descriptors extracted from training images. For this purpose, we first selected a random subset of training images (about 30 images per class), then we collected all SIFT descriptors of these images and run $k$-means. In all the experiments, we computed dictionaries of 500 visual words.

The results obtained with the three different settings are depicted in Fig. 12.7. Notice that UNN using the Histogram Intersection matching outperforms all the compared curves. We also note an improvement (up to 5 % gap for $k$-NN and 7 % for UNN) when using L1-normalized Bag-of-Features descriptors compared to Euclidean distance. This similarity measure was firstly proposed by [39] for image indexing based on color histograms, and, more recently, it has been successfully used by [23] in the context of Bag-of-Features image categorization.

[4]Code available at http://www.vlfeat.org/.

### 12.3.3 Comparison with SVM and AdaBoost on Image Categorization

Two major issues arise when implementing our UNN algorithm in practice. The first one concerns the distance (or, more generally, the *dissimilarity*) measure used for the $k$-NN search. The second one consists in setting the value of $k$ for both training and testing our prototype-based classifiers.

On the one hand, defining the most appropriate dissimilarity measure for $k$-NN search is particularly challenging when dealing with very high-dimensional feature vectors like image descriptors commonly used for categorization. Indeed, the classic metric distances may be inadequate when such vectors are generated by sophisticated pre-processing stages (e.g., vector quantization or unsupervised dictionary learning), thus lying on complex high-dimensional manifolds. In general, this should require an additional distance learning stage in order to define the optimal dissimilarity measure for the particular type of data at hand. In this respect, our UNN method has the advantage of being fully complementary with any metric learning algorithm, acting on the top of the $k$-NN search. In Sect. 12.3.2, we have described some examples of using different distances for $k$-NN search, particularly focusing on the most suitable dissimilarity measure for histogram-based descriptors.

On the other hand, selecting a good value for $k$ amounts to learning parameter-dependent weak classifiers, where the parameter $k$ specifies the size of the voting neighborhood in classification rule (12.10). From the theoretical standpoint, a brute-force approach is possible with boosting: one can define multiple candidate weak classifiers per example, one for each value of $k$, that is, for each neighborhood size, and then learn prototypes by optimizing the surrogate risk function over $k$ as well. This strategy has the advantage of enabling direct learning of $k$ at training time. However, training several weak classifiers per example without computation tricks would potentially severely impair the applicability of the algorithm on huge datasets. The solution we propose is subtler, as it relies on weighting the neighbors, exploiting the trick that boosting locally fits particular maximum likelihood estimators of class memberships [27]. Using (12.14), we can indeed rewrite (12.10) as:

$$h_c^{\ell}(\boldsymbol{x}) \approx \log \prod_{j \sim_k \boldsymbol{x}, y_{jc}>0} \frac{\hat{p}(c|j)}{\hat{p}(\bar{c}|j)} - \log \prod_{j \sim_k \boldsymbol{x}, y_{jc}<0} \frac{\hat{p}(c|j)}{\hat{p}(\bar{c}|j)}, \tag{12.24}$$

where $\hat{p}(c|j)$ (resp. $\hat{p}(\bar{c}|j)$) models a conditional probability (resp. not) to belong to class $c$. To make the right-hand side of (12.24) closer to a full-fledged maximum likelihood, we have to integrate the density estimators for nearest neighbors, $\hat{p}(j)$. We can obviously make the assumption that they are all equal: this would multiply the right-hand side of (12.24) by a positive constant factor, and would not change the outcome of (12.10). Instead, we have modified the classification phase of UNN, and tried a *soft* solution which considers a logistic estimator for a Bernoulli prior which vanishes with the rank of the example in the neighbors, thus decreasing the

importance of the farthest neighbors:

$$\hat{p}(j) = \beta_j = \frac{1}{1 + \exp(\lambda(j-1))}, \tag{12.25}$$

with $\lambda > 0$. The shape prior is chosen this way because it was shown that boosting, as carried out in a number of algorithms—not restricted to the induction of linear separators [27]—locally fits logistic estimators for Bernoulli priors. The soft version of UNN we obtain, called UNN$_s$ (for "Soft UNN"), replaces (12.10) by:

$$h_c^{\ell}(\boldsymbol{x}) = \sum_{j \sim_k \boldsymbol{x}} \beta_j \alpha_{jc} y_{jc}. \tag{12.26}$$

Notice that it is useless to enforce the normalization of coefficients $\beta_j$ in (12.25), because it would not change the classification of UNN$_s$. Notice also that the $\beta_j$s in (12.26) are used only to classify new observations: the training steps of UNN$_s$ are the same as UNN, and so UNN$_s$ meets the same theoretical properties as UNN described in Theorems 12.1, 12.2 and 12.3.

We selected 100 categories from the SUN database [43]. We kept all the images of each category and the inherent unbalancing of the original database. We randomly chose half images to form a training set, while testing on the remaining ones. The MAP was computed by averaging classification rates over categories (diagonal of the confusion matrix) and then averaging those values after repeating each experiment 10 times on different folds. To speed-up processing time, we used the fast implementation of $k$-NN proposed by [21].[5] Furthermore, we also developed an optimized version of our program, which exploits *multi-thread* functionalities. We denote this version as UNN$_s$(MT). All the experiments were run on an Intel Xeon X5690 12-cores processor at 3.46 GHz.

We compared UNN$_s$, SVM with Gaussian RBF Kernel, and AdaBoost with decision stumps[6] (i.e., decision trees with a single internal node), using BoF descriptors. In particular, we followed the guidelines of [20] for carrying out the SVM experiments, thus carrying out cross-validation for selecting the best parameters values for SVM. For the sake of completeness, we also provide results for Gist descriptors with UNN$_s$ and $k$-NN.

In Table 12.1, we report the MAP for each classification method. Results in these tables are provided as a function of the number of image categories. The most relevant results obtained are also displayed in Fig. 12.8 (mAP as a function of the number of categories) and Figs. 12.9 and 12.10, for the training and classification times, respectively.

[5] Code available at http://www.irisa.fr/texmex/people/jegou/src.php.

[6] For AdaBoost, we used the code available at http://www.mathworks.com/matlabcentral/fileexchange/22997-multiclass-gentleadaboosting.

**Table 12.1** Classification performances of the different methods we tested in terms of the Mean Average Precision (MAP) as a function of the number of categories

| # categories | 10 | 20 | 30 | 40 | 50 | 60 | 100 |
|---|---|---|---|---|---|---|---|
| $k$-NN BoF | 76.38 | 57.28 | 45.00 | 40.27 | 36.09 | 32.30 | 24.67 |
| SVM BoF | 83.85 | 67.65 | 58.21 | *53.45* | *47.81* | *44.09* | *35.31* |
| AdaBoost BoF | 75.37 | 58.21 | 45.57 | 37.75 | 32.41 | 29.01 | 26.72 |
| UNN$_s$ BoF | *84.28* | *70.44* | *58.49* | 51.07 | 46.34 | 41.80 | 31.61 |
| $k$-NN Gist | 64.22 | 51.48 | 43.65 | 39.04 | 35.65 | 32.27 | 25.50 |
| UNN$_s$ Gist | 77.84 | 66.80 | 56.37 | 50.45 | 46.48 | 42.75 | 32.71 |

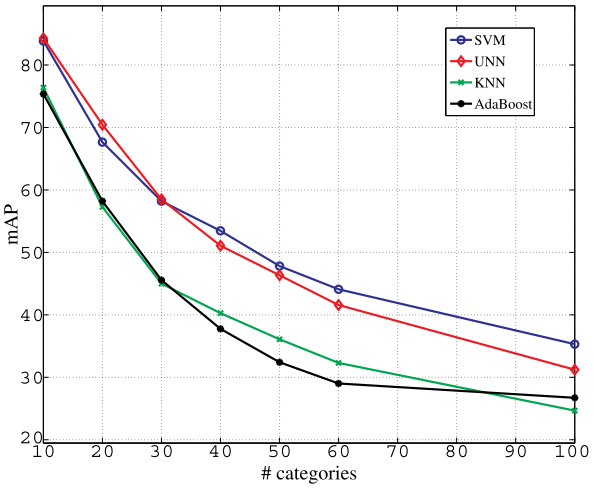


**Fig. 12.8** Classification performances of the tested methods as a function of the number of image categories

We can first notice that BoF descriptors generally outperform Gist, even when this phenomenon is dampened as the number of categories increases (above 30). This, overall, follows the trend generally reported in the literature.

MAP results display that UNN$_s$ dramatically outperforms AdaBoost (and $k$-NN as well); this result, which somehow experimentally confirms that UNN successfully exploits the boosting theory, was quite predictable, as UNN builds a piecewise linear decision function in the initial domain $\mathcal{O}$, while AdaBoost builds a linear separator in this domain. SVM, on the other hand, have access to non-linear fitting of data, by lifting the data to a domain whose dimension far exceeds that of $\mathcal{O}$. Yet, SVM's testing results are somehow not as good as one might expect from this clearcut theoretical advantage over UNN, and also from the fact that we carried out SVMs with significant parameters optimization [20]. Indeed, UNN$_s$ even beats SVMs over 10 to 30 categories, being slightly outperformed by them on more categories.

In Tables 12.2 and 12.3, we report the corresponding computation time (in seconds) for the training and classification phase, respectively. Obviously, the computation times over training and testing are also a key for exploiting the experimental re-

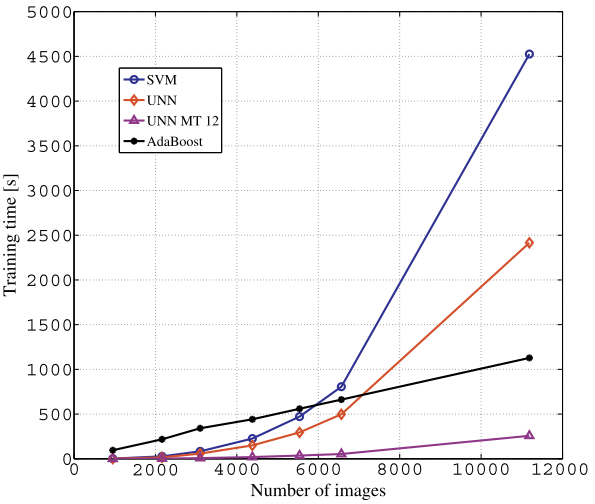


**Fig. 12.9** Training time as a function of the number of image categories

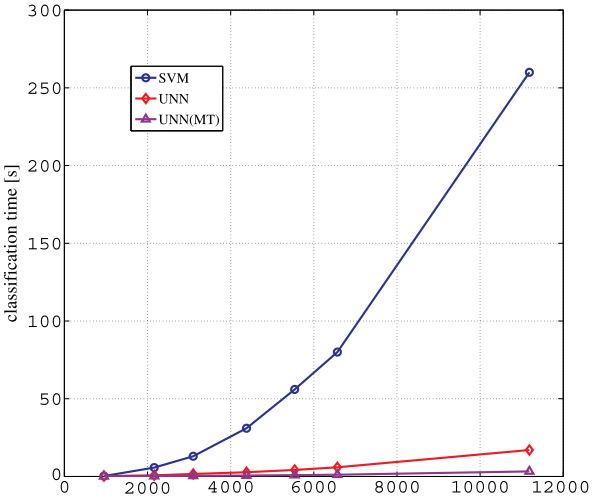


**Fig. 12.10** Classification time for $\text{UNN}_s$ vs SVM as a function of the number of image categories with BoF

sults. Table 12.2 displays that, while the training time of AdaBoost is linear, $\text{UNN}_s$ is a logical clearcut winner over SVM for training: it achieves speedups ranging in between two and more than seventeen over SVM. To assess the validity of these comparisons, we have computed least-square fittings of the training and testing times of $\text{UNN}_s$ vs AdaBoost vs SVM (all with BoF), with both linear ($s = aC + b$, $s$ being the time in $s$econds, and $C$ the number of categories) and polynomial ($s = bC^a$) fittings, with the objective to foresee on the best models what might happen on domains with classes ranging from hundreds to (tens of) thousands. The best models are displayed in Table 12.4. The coefficients of determination show that only a slim portion of the data is not explained by the models shown.

**Table 12.2** Computation time [s] for the training phase

| # categories | 10 | 20 | 30 | 40 | 50 | 60 | 100 |
|---|---|---|---|---|---|---|---|
| # training images | 951 | 2,162 | 3,099 | 4,381 | 5,540 | 6,568 | 11,186 |
| $k$-NN BoF | | | | 0 | | | |
| SVM BoF | 2.4 | 27 | 83 | 226 | 472 | 806 | 4526 |
| AdaBoost BoF | 96 | 218 | 341 | 442 | 559 | 662 | 1128 |
| $\text{UNN}_s$ BoF | 1.7 | 16 | 58 | 150 | 295 | 498 | 2146 |
| $\text{UNN}_s$(MT) BoF | 0.3 | 2.5 | 7.8 | 19 | 36 | 53 | 257 |

**Table 12.3** Computation time [s] for the testing phase

| # categories | 10 | 20 | 30 | 40 | 50 | 60 | 100 |
|---|---|---|---|---|---|---|---|
| # test images | 951 | 2,162 | 3,099 | 4,381 | 5,540 | 6,568 | 11,186 |
| $k$-NN BoF | 0.20 | 1.0 | 2.0 | 4.0 | 6.0 | 9.0 | 22.0 |
| SVM BoF | 0.25 | 5.7 | 13 | 31 | 56 | 80 | 260 |
| AdaBoost BoF | 0.02 | 0.1 | 0.25 | 0.43 | 0.67 | 0.95 | 2.74 |
| $\text{UNN}_s$ BoF | 0.21 | 0.72 | 1.6 | 2.7 | 4.2 | 5.9 | 17 |
| $\text{UNN}_s$(MT) BoF | 0.08 | 0.2 | 0.37 | 0.58 | 0.84 | 1.11 | 3.25 |

Models confirm that the training time of AdaBoost is linear. This is not a surprise, as it is ran with stumps as weak classifiers. Allowing decision trees with more than one internal node would have certainly blown the linear time barrier. While they are roughly equivalent for $\text{UNN}_s$ and AdaBoost (Table 12.3), testing times reveal a much bigger gap between $\text{UNN}_s$ and SVM, as displayed in Fig. 12.10. Exploiting the models of Table 12.4, we obtain the ratio:

$$s_{\text{SVM}}/s_{\text{UNN}} \approx \Omega(m), \tag{12.27}$$

while, for the multi-thread implementation, we obtain:

$$s_{\text{SVM}}/s_{\text{UNN}_s\text{MT}} \approx \Omega\left(m^{1.3}\right). \tag{12.28}$$

The ratio is always in favor of UNN, and of order the number of examples. Hence, the execution time for $\text{UNN}_s$ should allow to classify many images in reduced time compared to SVM: from Table 12.4, UNN should already classify almost twenty times as many images as SVM in a single minute. In such a case, $\text{UNN}_s$ should also classify almost twice as many images as AdaBoost. Thus, UNN provides the best MAP/time trade-off among the tested methods, which suggests that UNN might well be more than a legal contender to classification methods dealing with huge domains, or domains where the testing set is huge compared to the training set, which is the case, for instance, for cell classification in biological images [16]. Finally, we have only scratched experimental optimizations for UNN, and have not optimized

**Table 12.4** Best fits for training/testing times [s] as a function of the number of classes $C$, or the number of images $m$ in the training sample/to be tested. The model indicated is the best fit among models of the type $y = ax + b$ and $y = bx^a$, according to the coefficient of determination $r^2$. For all but two models, $r^2 > 0.999 \approx 1.0$ (the exceptions are (*), for which $r^2 \approx 0.97$, and (**), for which $r^2 \approx 0.99$). $m_{1\text{m}}$ is the number, estimated by the model, of images that can be processed in 1 minute (see text for details)

| | Training | | Testing | | |
|---|---|---|---|---|---|
| | $s = f(C)$ | $s = f(m)$ | $s = f(C)$ | $s = f(m)$ | $m_{1\text{m}}$ |
| SVM BoF | $s = (1.42 \times 10^{-3}) \times C^{3.25}$ | $s = (1.9 \times 10^{-9}) \times m^{3.05}$ | $s = (4.94 \times 10^{-4}) \times C^{2.94}$ (*) | $s = (2.11 \times 10^{-9}) \times m^{2.77}$ | 5 942 |
| AdaBoost BoF | $s = -11.40 + 11.37 \times C$ | $s = 7.63 + 0.10 \times m$ | $s = (2.16 \times 10^{-4}) \times C^{2.05}$ | $s = (4.19 \times 10^{-8}) \times m^{1.93}$ | 55 643 |
| UNN$_s$ BoF | $s = (1.24 \times 10^{-3}) \times C^{3.16}$ | $s = (2.43 \times 10^{-9}) \times m^{2.96}$ | $s = (2.49 \times 10^{-3}) \times C^{1.90}$ | $s = (9.19 \times 10^{-7}) \times m^{1.78}$ | 24 567 |
| UNN$_s$ MT BoF | $s = (3.85 \times 10^{-4}) \times C^{2.91}$ | $s = (2.07 \times 10^{-9}) \times m^{2.74}$ | $s = (1.82 \times 10^{-3}) \times C^{1.58}$ | $s = (2.57 \times 10^{-6}) \times m^{1.48}$ (**) | 95 175 |

UNN from the complexity-theoretic standpoint, so we expect room space for further significant improvement of its training/testing times.

## 12.4 Discussion and Perspectives

UNN provides us with a sound blend of two powerful yet simple classification algorithms: nearest neighbors and boosting. While the analysis of the mixing is not straightforward—such as for the convergence and boosting properties in Theorems 12.1–12.3—UNN remains simple to state and implement, even in the multiclass case. It also appears to be a interesting contender to SVM: without using the kernel trick mapping examples to high dimensional feature spaces, UNN manages to fit nonlinear classifiers in the initial feature space whose accuracy clearly compete with SVM's.

We think that this simplicity opens avenues for future research on the way separate extensions and improvements of nearest neighbors and boosting might be transferred to UNN. One example is the inclusion of powerful density estimation techniques that would fit better than our simple logistic convolution of priors in (12.25).

Another example involves improved sophistication from the classifier's standpoint, in particular with metric distance learning and the kernelization of the input space [47]. This, we expect, would enable significant improvements of categorization performances.

A third example involves improvements from the nearest neighbor search standpoint. Novel techniques exist that make embeddings in a real-valued vector space of nearest neighbors queries, thus transforming the data space with the hope to achieve good compromises between reducing the processing complexity of nearest neighbor queries while not reducing the accuracy of (vanilla) nearest neighbors in the space learnt [2, 25]. Clearly, such approaches do not tackle the same problem as us, as UNN directly processes nearest neighbors on the data's ambient space. Nevertheless, they are very interesting from the perspective standpoint because this new data space is learnt with (Ada)boosting. A neat combination with UNN might thus offer the possibility to kill two birds in one boosting shot for nearest neighbors: learn an improved data space, *and* learn in this data space an improved nearest neighbor classifier with UNN. The questions raised by such perspective are not only experimental, as basically only the contractiveness of the approach of [2] is formally known to date. Transferring, or even improving, the boosting properties of UNN in such sophisticated blends would thus be more than interesting.

## 12.5 Conclusion

In this work, we contribute to fill an important void of NN methods, showing how boosting can be transferred to $k$-NN classification, with convergence rates guarantees for a *large* number of surrogates. Our UNN algorithm generalizes classic $k$-NN

to weighted voting where weights, the so-called leveraging coefficients, are iteratively learned by UNN. We prove that this algorithm converges to the global optimum of many surrogate risks in competitive times under very mild assumptions.

Our work is also the first extensive assessment of UNN to computer vision related tasks. Comparisons with $k$-NN, support vector machines and AdaBoost, using Gist or Bag-of-Feature descriptors, on simulated and real domains, display the ability of UNN to be competitive with its contenders, achieving high mAP in comparatively reduced training and testing times.

Avenues for future research include blending UNN with other approaches that bias the domain towards the improvement of nearest neighbors rules, or that learn more sophisticated metrics over data.

## Appendix

**Generic UNN Algorithm** The general version of UNN is shown in Algorithm 2. This algorithm induces the leveraged $k$-NN rule (12.10) for the broad class of surrogate losses meeting conditions of [4], thus generalizing Algorithm 1. Namely, we constrain $\psi$ to meet the following conditions: (i) $\text{im}(\psi) = \mathbb{R}_+$, (ii) $\nabla_\psi(0) < 0$ ($\nabla_\psi$ is the conventional derivative of $\psi$ loss function), and (iii) $\psi$ is strictly convex and differentiable. (i) and (ii) imply that $\psi$ is *classification-calibrated*: its local minimization is roughly tied up to that of the empirical risk [4]. (iii) implies convenient algorithmic properties for the minimization of the surrogate risk [28]. Three common examples have been shown in Eqs. (12.7)–(12.6).

The main bottleneck of UNN is step [I.1], as Eq. (12.30) is non-linear, *but* it always has a solution, finite under mild assumptions [28]: in our case, $\delta_j$ is guaranteed to be finite when there is no total matching or mismatching of example $j$'s memberships with its reciprocal neighbors', for the class at hand. The second column of Table 12.5 contains the solutions to (12.30) for surrogate losses mentioned in Sect. 12.2.2. Those solutions are always exact for the exponential loss ($\psi^{\text{exp}}$) and squared loss ($\psi^{\text{squ}}$); for the logistic loss ($\psi^{\log}$) it is exact when the weights in the reciprocal neighborhood of $j$ are the same, otherwise it is approximated. Since starting weights are all the same, exactness can be guaranteed during a large number of inner rounds depending on which order is used to choice the examples. Table 12.5 helps to formalize the finiteness condition on $\delta_j$ mentioned above: when either sum of weights in (12.29) is zero, the solutions in the first and third line of Table 12.5 are not finite. A simple strategy to cope with numerical problems arising from such situations is that proposed by [35]. (See Sect. 12.2.4.) Table 12.5 also shows how the weight update rule (12.31) specializes for the mentioned losses.

**Proofsketch of Theorem 12.1** We show that UNN converges to the global optimum of any surrogate risk (Sect. 12.2.5). For this purpose, let us consider the

**Algorithm 2** Universal Nearest Neighbors UNN($\mathcal{S}, \psi$)

**Input:** $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), \quad i = 1, 2, \ldots, m, \quad \boldsymbol{x}_i \in \mathcal{X}, \quad \boldsymbol{y}_i \in \{-\frac{1}{C-1}, 1\}^C\}$, $\psi$ meeting (i), (ii), (iii) (Appendix);

$\mathrm{r}_{ij}^{(c)} \doteq \begin{cases} y_{ic} y_{jc} & \text{if } j \sim_k i \\ 0 & \text{otherwise} \end{cases} \quad \forall i, j = 1, 2, \ldots, m, \ c = 1, 2, \ldots, C$ ▷ $k$-NN edge matrix

**for** $c = 1, 2, \ldots, C$ **do**

$\alpha_{jc} \leftarrow 0, \quad \forall j = 1, 2, \ldots, m$ ▷ Leveraging coefficients

$w_i \leftarrow -\nabla_\psi(0) \in \mathrm{R}^m_{+*}, \quad \forall i = 1, 2, \ldots, m$ ▷ Example weights

**for** $t = 1, 2, \ldots, T$ **do**

**[I.0]** $j \leftarrow \textsc{Wic}(\{1, 2, \ldots, m\}, t)$ ▷ Weak index chooser oracle

**[I.1]**

$$w_j^+ = \sum_{i:\mathrm{r}_{ij}^{(c)} > 0} w_i, \qquad w_j^- = \sum_{i:\mathrm{r}_{ij}^{(c)} < 0} w_i, \tag{12.29}$$

Let $\delta_j \in \mathbb{R}$ solution of:

$$\sum_{i=1}^m \mathrm{r}_{ij}^{(c)} \nabla_\psi\left(\delta_j \mathrm{r}_{ij}^{(c)} + \nabla_\psi^{-1}(-w_i)\right) = 0; \tag{12.30}$$

**[I.2]** $\forall i : \ j \sim_k i$, let

$$w_i \leftarrow -\nabla_\psi\left(\delta_j \mathrm{r}_{ij}^{(c)} + \nabla_\psi^{-1}(-w_i)\right). \tag{12.31}$$

**[I.3]** $\alpha_{jc} \leftarrow \alpha_{jc} + \delta_j$

**end for**

**end for**

**Output:** $h_c(\boldsymbol{x}_{i'}) = \sum_{i \sim_k i'} \alpha_{ic} y_{ic}, \quad \forall c = 1, 2, \ldots, C$

**Table 12.5** Three common loss functions and the corresponding solutions $\delta_j$ of (12.30) and $w_i$ of (12.31). (Vector $\boldsymbol{r}_j^{(c)}$ designates column $j$ of $\mathrm{R}^{(c)}$ and $\|.\|_1$ is the $L_1$ norm.) The rightmost column says whether it is (A)lways the solution, or whether it is when the weights of reciprocal neighbors of $j$ are the (S)ame

| Loss function | $\delta_j$ in (12.30) | $w_i$ in (12.31) | Opt |
|---|---|---|---|
| $\psi^{\exp} \doteq \exp(-x)$ | $\frac{1}{2}\log(\frac{w_j^{(c)+}}{w_j^{(c)-}})$ | $w_i \exp\left(-\delta_j \mathrm{r}_{ij}^{(c)}\right)$ | 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 |
| $\psi^{\log} \doteq \log(1+\exp(-x))$ | $\log(\frac{w_j^{(c)+}}{w_j^{(c)-}})$ | $\frac{w_i \exp(-\delta_j \mathrm{r}_{ij}^{(c)})}{1 - w_i(1+\exp(-\delta_j \mathrm{r}_{ij}^{(c)}))}$ | S |

surrogate risk (12.5) for a given class $c = 1, 2, \ldots, C$:

$$\varepsilon_c^\psi(\boldsymbol{h}, \mathcal{S}) \doteq \frac{1}{m} \sum_{i=1}^m \psi\left(\varrho(\boldsymbol{h}, i, c)\right). \tag{12.32}$$

In this section, we use 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 $\psi$, which is strictly convex and differentiable as well. ($\tilde{\psi}$ is related to $\psi$ in such a way that: $\nabla_{\tilde{\psi}}(x) = -\nabla_\psi^{-1}(-x)$.)
- $D_{\tilde{\psi}}(w_i \| w_i') \doteq \tilde{\psi}(w_i) - \tilde{\psi}(w_i') - (w_i - w_i')\nabla_{\tilde{\psi}}(w_i')$ is the Bregman divergence with generator $\tilde{\psi}$ [28].

Let $\boldsymbol{w}_t$ denote the $t$th weight vector inside the "**for** $c$" loop of Algorithm 2 (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 fundamental identity holds, whose proof follows from [28]:

$$\psi\big(\varrho\big(\boldsymbol{h}_t^\ell, i, c\big)\big) = g + D_{\tilde{\psi}}(0 \| w_{ti}), \tag{12.33}$$

where $g(m) \doteq -\tilde{\psi}(0)$ does not depend on the $k$-NN rule. In particular, Eq. (12.33) makes the connection between the real-valued classification problem and a geometric problem in the non-metric space of weights. Moreover, Eq. (12.33) proves in handy as one computes the *difference* between two successive surrogates: $\varepsilon_c^\psi(\boldsymbol{h}_{t+1}^\ell, \mathcal{S}) - \varepsilon_c^\psi(\boldsymbol{h}_t^\ell, \mathcal{S})$. Indeed, plugging Eq. (12.33) in Eq. (12.32), and computing $\delta_j$ in Eq. (12.30) so as to bring $\boldsymbol{h}_{t+1}^\ell$ from $\boldsymbol{h}_t^\ell$, we obtain the following identity:

$$\varepsilon_c^\psi\big(\boldsymbol{h}_{t+1}^\ell, \mathcal{S}\big) - \varepsilon_c^\psi\big(\boldsymbol{h}_t^\ell, \mathcal{S}\big) = -\frac{1}{m}\sum_{i=1}^m D_{\tilde{\psi}}(w_{(t+1)i} \| w_{ti}). \tag{12.34}$$

Since Bregman divergences are non negative and meet the identity of the indiscernibles, (12.34) implies that steps [I.1]–[I.3] *guarantee* the decrease of (12.32) as long as $\delta_j \neq 0$. But (12.32) is lowerbounded, hence UNN must converge.

In addition, it converges to the global optimum of the risk (12.5). Since predictions for each class are independent, the proof consists in showing that (12.32) converges to its global minimum for each $c$. Let us assume this convergence for the current class $c$. Then, following the reasoning of Nock and Nielsen [28], (12.30) and (12.31) imply that, when any possible $\delta_j = 0$, the weight vector, say $\boldsymbol{w}_\infty$, satisfies ${\mathrm{R}^{(c)}}^\top \boldsymbol{w}^\top = \boldsymbol{0}$, *that is*, $\boldsymbol{w}_\infty \in \ker {\mathrm{R}^{(c)}}^\top$, and $\boldsymbol{w}_\infty$ is unique. But the kernel of ${\mathrm{R}^{(c)}}^\top$ and $\overline{\mathbb{W}}$, the closure of $\mathbb{W}$ (i.e., the manifold where $\boldsymbol{w}$'s live), are provably Bregman orthogonal [28], thus yielding:

$$\underbrace{\sum_{i=1}^m D_{\tilde{\psi}}(0 \| w_i)}_{m\varepsilon_c^\psi(\boldsymbol{h}^\ell, \mathcal{S}) - mg} = \underbrace{\sum_{i=1}^m D_{\tilde{\psi}}(0 \| w_{\infty i})}_{m\varepsilon_c^\psi(\boldsymbol{h}_\infty^\ell, \mathcal{S}) - mg} + \underbrace{\sum_{i=1}^m D_{\tilde{\psi}}(w_{\infty i} \| w_i)}_{\geq 0}, \quad \forall \boldsymbol{w} \in \overline{\mathbb{W}}. \tag{12.35}$$

Underbraces use (12.33) in (12.32), 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 (12.32), as claimed.

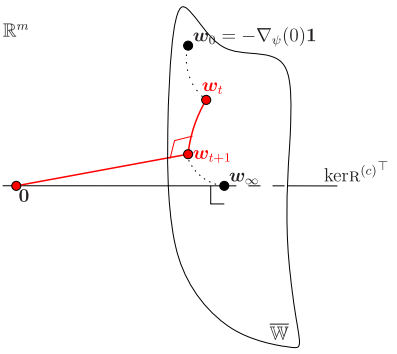


**Fig. 12.11** A geometric view of how UNN converges to the global optimum of (12.5). (See Appendix for details and notations.)

The proofsketch is graphically summarized in Fig. 12.11. In particular, two crucial *Bregman orthogonalities* are mentioned [28]. The red one symbolizes:

$$\sum_{i=1}^{m} D_{\tilde{\psi}}(0 \| w_{ti}) = \sum_{i=1}^{m} D_{\tilde{\psi}}(0 \| w_{(t+1)i}) + \sum_{i=1}^{m} D_{\tilde{\psi}}(w_{(t+1)i} \| w_{ti}), \tag{12.36}$$

which is equivalent to (12.34). The black one on $\boldsymbol{w}_\infty$ is (12.35).

**Proofsketch of Theorem 12.2** Using developments analogous to those of [28], 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 [35] brings in our case:

$$\varepsilon^{0/1}(\boldsymbol{h}^\ell, \mathcal{S}) \leq \frac{1}{C} \sum_{c=1}^{C} \prod_{t=1}^{T} Z_t^{(c)}, \tag{12.37}$$

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:

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

where $\tilde{w}_{jt}^{(c)+-} \doteq \tilde{w}_{jt}^{(c)+} + \tilde{w}_{jt}^{(c)-}$, $p_{jt}^{(c)} \doteq \tilde{w}_{jt}^{(c)+} / \tilde{w}_{jt}^{(c)+-} = w_{jt}^{(c)+} / w_{jt}^{(c)+-}$. The first inequality 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 (12.37) yields the statement of the theorem.

**Proofsketch of Theorem 12.3** We plug in the weight notation the iteration $t$ and class $c$, so that $w_{ti}^{(c)}$ denotes the weight of example $\boldsymbol{x}_i$ prior to iteration $t$ for class $c$ in UNN (inside the "**for** $c$" loop of Algorithm 2, letting $\boldsymbol{w}_0$ denote the initial value of $\boldsymbol{w}$). To save space in some computations below, we also denote for short:

$$\bar{\varepsilon}^{\psi}\left(\boldsymbol{h}_T^{\ell}, \mathcal{S}\right) \doteq \frac{1}{C}\sum_{c=1}^{C} \varepsilon_c^{\psi}\left(\boldsymbol{h}_T^{\ell}, \mathcal{S}\right). \tag{12.38}$$

$\psi$ is $\omega$ strongly smooth is equivalent to $\tilde{\psi}$ being strongly convex with parameter $\omega^{-1}$ [22], that is,

$$\tilde{\psi}(w) - \frac{1}{2\omega}w^2 \tag{12.39}$$

is convex. 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 $\psi$. Since a convex function $h$ satisfies $h(w') \geq h(w) + \nabla_h(w)(w' - w)$, applying inequality (12.39) taking as $h$ the function in (12.39) yields, $\forall t = 1, 2, \ldots, T$, $\forall i = 1, 2, \ldots, m$, $\forall c = 1, 2, \ldots, C$:

$$\begin{aligned} D_{\tilde{\psi}}\left(w_{(t+1)i}^{(c)} \| w_{ti}^{(c)}\right) &= D_{\tilde{\psi}}\left(w_{ti}^{(c)} + \left(w_{(t+1)i}^{(c)} - w_{ti}^{(c)}\right) \| w_{ti}^{(c)}\right) \\ &\geq \frac{1}{2\omega}\left(w_{(t+1)i}^{(c)} - w_{ti}^{(c)}\right)^2, \end{aligned} \tag{12.40}$$

where we recall that $D_{\psi}$ denotes the Bregman divergence with generator $\psi$ (12.22). On the other hand, Cauchy–Schwarz inequality yields:

$$\begin{aligned} \forall j \in \mathcal{S}, \quad \sum_{i:j\sim_k i} \left(\mathrm{r}_{ij}^{(c)}\right)^2 \sum_{i:j\sim_k i} \left(w_{(t+1)i}^{(c)} - w_{ti}^{(c)}\right)^2 &\geq \left(\sum_{i:j\sim_k i} \mathrm{r}_{ij}^{(c)}\left(w_{(t+1)i}^{(c)} - w_{ti}^{(c)}\right)\right)^2 \\ &= \left(\sum_{i:j\sim_k i} \mathrm{r}_{ij}^{(c)} w_{ti}^{(c)}\right)^2. \end{aligned} \tag{12.41}$$

The equality in (12.41) holds because $\sum_{i:j\sim_k i} \mathrm{r}_{ij}^{(c)} w_{(t+1)i}^{(c)} = 0$, which is exactly (12.30). We obtain:

$$\begin{aligned} \frac{1}{m}\sum_{i=1}^{m} D_{\tilde{\psi}}\left(w_{(t+1)i}^{(c)} \| w_{ti}^{(c)}\right) &= \frac{1}{m}\sum_{i:t\sim_k i} D_{\tilde{\psi}}\left(w_{(t+1)i}^{(c)} \| w_{ti}^{(c)}\right) \\ &\geq \frac{1}{2\omega m}\sum_{i:t\sim_k i}\left(w_{(t+1)i}^{(c)} - w_{ti}^{(c)}\right)^2 \end{aligned} \tag{12.42}$$

$$\geq \frac{1}{2\omega m} \frac{(\sum_{i:t\sim_k i} \mathrm{r}_{it}^{(c)} w_{ti}^{(c)})^2}{\sum_{i:t\sim_k i} (\mathrm{r}_{it}^{(c)})^2} \tag{12.43}$$

$$\geq \frac{\vartheta^2}{2\omega m} \times \frac{1}{\sum_{i:t\sim_k i} (\mathrm{r}_{it}^{(c)})^2} \tag{12.44}$$

Here, (12.42) follows from (12.40), (12.43) follows from (12.41), and (12.44) follows from (12.20). Adding (12.44) for $c = 1, 2, \ldots, C$ and $t = 1, 2, \ldots, T$, and then dividing by $C$, we obtain:

$$\frac{1}{C} \sum_{c=1}^{C} \sum_{t=1}^{T} \frac{1}{m} \sum_{i=1}^{m} D_{\tilde{\psi}} \left( w_{(t+1)i}^{(c)} \| w_{ti}^{(c)} \right)$$

$$\geq \frac{T\vartheta^2}{2\omega m} \times \left( \frac{1}{TC} \times \sum_{c=1}^{C} \sum_{t=1}^{T} \frac{1}{\sum_{i:t\sim_k i} (\mathrm{r}_{it}^{(c)})^2} \right). \tag{12.45}$$

We now work on the big parenthesis which depends solely upon the examples. We have:

$$\left( \frac{1}{TC} \times \sum_{c=1}^{C} \sum_{t=1}^{T} \frac{1}{\sum_{i:t\sim_k i} (\mathrm{r}_{it}^{(c)})^2} \right)^{-1}$$

$$\leq \frac{1}{TC} \sum_{c=1}^{C} \sum_{t=1}^{T} \sum_{i:t\sim_k i} \left( \mathrm{r}_{it}^{(c)} \right)^2 \tag{12.46}$$

$$= \frac{1}{TC} \sum_{c=1}^{C} \sum_{t=1}^{T} \sum_{i \in \mathrm{NN}_k(\boldsymbol{x}_t)} y_{tc}^2 y_{ic}^2$$

$$\leq \frac{1}{TC} \sum_{c=1}^{C} \sum_{t=1}^{T} \sum_{i \in \mathrm{NN}_k(\boldsymbol{x}_t)} \left( \frac{|y_{tc}|}{2} + \frac{|y_{ic}|}{2} \right) \tag{12.47}$$

$$= \frac{k}{TC} \sum_{t=1}^{T} \sum_{c=1}^{C} \frac{|y_{tc}|}{2} + \frac{1}{TC} \sum_{t=1}^{T} \sum_{i \in \mathrm{NN}_k(\boldsymbol{x}_t)} \sum_{c=1}^{C} \frac{|y_{ic}|}{2}$$

$$= \frac{k}{(C-1)}. \tag{12.48}$$

Here, (12.46) holds because of the Arithmetic-Geometric-Harmonic inequality, and (12.47) is Young's inequality[7] with $p = q = 2$. Plugging (12.48) into (12.45), we obtain:

[7] We recall young inequality: for any $p, q$ Hölder conjugates ($p > 1$, $(1/p) + (1/q) = 1$), we have $yy' \leq y^p/p + y'^q/q$, assuming $y, y' \geq 0$.

$$\frac{1}{C}\sum_{c=1}^{C}\sum_{t=1}^{T}\frac{1}{m}\sum_{i=1}^{m} D_{\tilde{\psi}}\left(w_{(t+1)i}^{(c)} \| w_{ti}^{(c)}\right) \geq \frac{T(C-1)\vartheta^2}{2\omega m k}. \tag{12.49}$$

Now, UNN meets the following property, which can easily be shown to hold with our class encoding as well:

$$\varepsilon_c^{\psi}\left(\boldsymbol{h}_{t+1}^{\ell}, \mathcal{S}\right) - \varepsilon_c^{\psi}\left(\boldsymbol{h}_{t}^{\ell}, \mathcal{S}\right) = -\frac{1}{m}\sum_{i=1}^{m} D_{\tilde{\psi}}\left(w_{(t+1)i}^{(c)} \| w_{ti}^{(c)}\right). \tag{12.50}$$

Adding (12.50) for $t = 0, 2, \ldots, T-1$ and $c = 1, 2, \ldots, C$, we obtain:

$$\frac{1}{C}\sum_{c=1}^{C}\varepsilon_c^{\psi}\left(\boldsymbol{h}_{T}^{\ell}, \mathcal{S}\right) - \psi(0) = -\frac{1}{C}\sum_{c=1}^{C}\sum_{t=1}^{T}\frac{1}{m}\sum_{i=1}^{m} D_{\tilde{\psi}}\left(w_{(t+1)i}^{(c)} \| w_{ti}^{(c)}\right). \tag{12.51}$$

Plugging (12.49) into (12.51), we obtain:

$$\bar{\varepsilon}^{\psi}\left(\boldsymbol{h}_{T}^{\ell}, \mathcal{S}\right) \leq \psi(0) - \frac{T(C-1)\vartheta^2}{2\omega m k}. \tag{12.52}$$

But the following inequality holds between the average surrogate risk and the empirical risk of the leveraged $k$-NN rule $\boldsymbol{h}_T^{\ell}$, because of (i):

$$\begin{aligned}
\bar{\varepsilon}^{\psi}\left(\boldsymbol{h}_{T}^{\ell}, \mathcal{S}\right) &= \frac{1}{C}\sum_{c=1}^{C}\varepsilon_c^{\psi}\left(\boldsymbol{h}_{T}^{\ell}, \mathcal{S}\right) \\
&= \frac{1}{mC}\sum_{c=1}^{C}\sum_{i=1}^{m}\psi\left(y_{ic}\sum_{j:j\sim_k i}\alpha_{jc}y_{jc}\right) \\
&\geq \frac{\psi(0)}{mC}\sum_{c=1}^{C}\sum_{i=1}^{m}\left[y_{ic}\sum_{j:j\sim_k i}\alpha_{jc}y_{jc} < 0\right] \\
&= \psi(0)\varepsilon^{0/1}\left(\boldsymbol{h}_{T}^{\ell}, \mathcal{S}\right),
\end{aligned} \tag{12.53}$$

so that, putting altogether (12.52) and (12.53) and using the fact that $\psi(0) > 0$ because of (i)–(ii), we have after $T$ rounds of boosting for each class: *that is*,

$$\varepsilon^{0/1}\left(\boldsymbol{h}_{T}^{\ell}, \mathcal{S}\right) \leq 1 - \frac{T(C-1)\vartheta^2}{2\psi(0)\omega m k}. \tag{12.54}$$

There remains to compute the minimal value of $T$ for which the right-hand side of (12.54) becomes no greater than some user-fixed $\tau \in [0, 1]$ to obtain the bound in (12.23).

## References


1. Amores J, Sebe N, Radeva P (2006) Boosting the distance estimation: application to the $k$-nearest neighbor classifier. Pattern Recognit Lett 27(3):201–209
2. Athitsos V, Alon J, Sclaroff S, Kollios G (2008) BoostMap: an embedding method for efficient nearest neighbor retrieval. IEEE Trans Pattern Anal Mach Intell 30(1):89–104
3. Bartlett P, Traskin M (2007) Adaboost is consistent. J Mach Learn Res 8:2347–2368
4. Bartlett P, Jordan M, McAuliffe JD (2006) Convexity, classification, and risk bounds. J Am Stat Assoc 101:138–156
5. Boutell MR, Luo J, Shen X, Brown CM (2004) Learning multi-label scene classification. Pattern Recognit 37(9):1757–1771
6. Brighton H, Mellish C (2002) Advances in instance selection for instance-based learning algorithms. Data Min Knowl Discov 6:153–172
7. Cucala L, Marin JM, Robert CP, Titterington DM (2009) A bayesian reassessment of nearest-neighbor classification. J Am Stat Assoc 104(485):263–273
8. Dudani S (1976) The distance-weighted $k$-nearest-neighbor rule. IEEE Trans Syst Man Cybern 6(4):325–327
9. Escolano Ruiz F, Suau Pérez P, Bonev BI (2009) Information theory in computer vision and pattern recognition. Springer, London
10. Fei-Fei L, Perona P (2005) A bayesian hierarchical model for learning natural scene categories. In: IEEE computer society conference on computer vision and pattern recognition (CVPR), pp 524–531
11. Fukunaga K, Flick T (1984) An optimal global nearest neighbor metric. IEEE Trans Pattern Anal Mach Intell 6(3):314–318
12. García-Pedrajas N, Ortiz-Boyer D (2009) Boosting $k$-nearest neighbor classifier by means of input space projection. Expert Syst Appl 36(7):10,570–10,582
13. Gionis A, Indyk P, Motwani R (1999) Similarity search in high dimensions via hashing. In: Proc international conference on very large databases, pp 518–529
14. Grauman K, Darrell T (2005) The pyramid match kernel: discriminative classification with sets of image features. In: IEEE international conference on computer vision (ICCV), pp 1458–1465
15. Gupta L, Pathangay V, Patra A, Dyana A, Das S (2007) Indoor versus outdoor scene classification using probabilistic neural network. EURASIP J Appl Signal Process 2007(1): 123
16. Bel Haj Ali W, Piro P, Crescence L, Giampaglia D, Ferhat O, Darcourt J, Pourcher T, Barlaud M (2012) Changes in the subcellular localization of a plasma membrane protein studied by bioinspired UNN learning classification of biologic cell images. In: International conference on computer vision theory and applications (VISAPP)
17. Hart PE (1968) The condensed nearest neighbor rule. IEEE Trans Inf Theory 14:515–516
18. Hastie T, Tibshirani R (1996) Discriminant adaptive nearest neighbor classification. IEEE Trans Pattern Anal Mach Intell 18(6):607–616
19. Holmes CC, Adams NM (2003) Likelihood inference in nearest-neighbour classification models. Biometrika 90:99–112
20. Hsu CW, Chang CC, Lin CJ (2003) A practical guide to support vector classification. Technical report
21. Jégou H, Douze M, Schmid C (2011) Product quantization for nearest neighbor search. IEEE Trans Pattern Anal Mach Intell 33(1):117–128
22. Kakade S, Shalev-Shwartz S, Tewari A (2009) Applications of strong convexity–strong smoothness duality to learning with matrices. Technical report
23. Lazebnik S, Schmid C, Ponce J (2006) Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: IEEE computer society conference on computer vision and pattern recognition (CVPR), pp 2169–2178
24. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110

25. Masip D, Vitrià J (2006) Boosted discriminant projections for nearest neighbor classification. Pattern Recognit 39(2):164–170
26. Nguyen X, Wainwright MJ, Jordan MI (2009) On surrogate loss functions and $f$-divergences. Ann Stat 37:876–904
27. Nock R, Nielsen F (2009) Bregman divergences and surrogates for learning. IEEE Trans Pattern Anal Mach Intell 31(11):2048–2059
28. Nock R, Nielsen F (2009) On the efficient minimization of classification calibrated surrogates. In: Advances in neural information processing systems (NIPS), vol 21, pp 1201–1208
29. Nock R, Sebban M (2001) An improved bound on the finite-sample risk of the nearest neighbor rule. Pattern Recognit Lett 22(3/4):407–412
30. Oliva A, Torralba A (2001) Modeling the shape of the scene: a holistic representation of the spatial envelope. Int J Comput Vis 42(3):145–175
31. Paredes R (2006) Learning weighted metrics to minimize nearest-neighbor classification error. IEEE Trans Pattern Anal Mach Intell 28(7):1100–1110
32. Payne A, Singh S (2005) Indoor vs. outdoor scene classification in digital photographs. Pattern Recognit 38(10):1533–1545
33. Piro P, Nock R, Nielsen F, Barlaud M (2012) Leveraging $k$-NN for generic classification boosting. Neurocomputing 80:3–9
34. Quattoni A, Torralba A (2009) Recognizing indoor scenes. In: IEEE computer society conference on computer vision and pattern recognition (CVPR)
35. Schapire RE, Singer Y (1999) Improved boosting algorithms using confidence-rated predictions. Mach Learn J 37:297–336
36. Serrano N, Savakis AE, Luo JB (2004) Improved scene classification using efficient low-level features and semantic cues. Pattern Recognit 37:1773–1784
37. Shakhnarovich G, Darell T, Indyk P (2006) Nearest-neighbors methods in learning and vision. MIT Press, Cambridge
38. Sivic J, Zisserman A (2003) Video google: a text retrieval approach to object matching in videos. In: IEEE international conference on computer vision (ICCV), vol 2, pp 1470–1477
39. Swain MJ, Ballard DH (1991) Color indexing. Int J Comput Vis 7:11–32
40. Torralba A, Murphy K, Freeman W, Rubin M (2003) Context-based vision system for place and object recognition. In: IEEE international conference on computer vision (ICCV), pp 273–280
41. Vedaldi A, Fulkerson B (2008) VLFeat: an open and portable library of computer vision algorithms. http://www.vlfeat.org
42. Vogel J, Schiele B (2007) Semantic modeling of natural scenes for content-based image retrieval. Int J Comput Vis 72(2):133–157
43. Xiao J, Hays J, Ehinger KA, Oliva A, Torralba A (2010) SUN database: large-scale scene recognition from abbey to zoo. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 3485–3492
44. Yu K, Ji L, Zhang X (2002) Kernel nearest-neighbor algorithm. Neural Process Lett 15(2):147–156
45. Yuan M, Wegkamp M (2010) Classification methods with reject option based on convex risk minimization. J Mach Learn Res 11:111–130
46. Zhang ML, Zhou ZH (2007) ML-kNN: a lazy learning approach to multi-label learning. Pattern Recognit 40(7):2038–2048
47. Zhang H, Berg AC, Maire M, Malik J (2006) SVM-kNN: discriminative nearest neighbor classification for visual category recognition. In: IEEE computer society conference on computer vision and pattern recognition (CVPR), pp 2126–2136

48. Zhu J, Rosset S, Zou H, Hastie T (2009) Multi-class adaboost. Stat Interface 2:349–360
49. Zuo W, Zhang D, Wang K (2008) On kernel difference-weighted $k$-nearest neighbor classification. Pattern Anal Appl 11(3–4):247–257