Learning prototype-based classification rules in a boosting framework: application to real-world and medical image categorization

Paolo Piro

Université de Nice - Sophia Antipolis

ÉCOLE DOCTORALE STIC

Sciences et Technologies de l'Information et de la Communication

THÈSE

pour obtenir le titre de

Docteur en Sciences

de l'Université de Nice - Sophia Antipolis

Mention : Automatique, Traitement du Signal et des Images

présentée et soutenue par

Paolo PIRO

Laboratoire Informatique, Signaux et Systèmes Sophia Antipolis

---

LEARNING PROTOTYPE-BASED CLASSIFICATION RULES IN A BOOSTING FRAMEWORK: APPLICATION TO REAL-WORLD AND MEDICAL IMAGE CATEGORIZATION

Thèse dirigée par Michel BARLAUD

Soutenue publiquement le 18 janvier 2010 devant le jury composé de

Cordelia SCHMID	Directeur de recherche INRIA	Présidente
Luigi CORDELLA	Professeur des Universités (Naples)	Rapporteur
Patrick PÉREZ	Directeur de recherche INRIA	Rapporteur
Francesco TORTORELLA	Professeur des Universités (Cassino)	Examinateur
Giulio IANNELLO	Professeur des Universités (Campus Bio-Medico-Rome)	Directeur en co-tutelle
Michel BARLAUD	Professeur des Universités (Nice-Sophia Antipolis) Membre de l'Institut Universitaire de France	Directeur de thèse

To my parents

"Et pour que je ne me décourageasse pas,
elle m'assura que du jour où je serais bien portant,
le travail viendrait tout seul par surcroît."
M. PROUST

CONTENTS

ACKNOWLEDGEMENTS

This PhD thesis has been a long and amazing excursion in a very attractive scientific field. I feel grateful to all the persons who have guided me along this way.
Many thanks to Prof. Michel Barlaud for advising me with great constancy and fervor. Thanks to Frank Nielsen and Richard Nock for their inestimable collaboration. Thanks to Prof. Giulio Iannello for his warm advice. Also thanks to Eric Debreuve and Sandrine Anthoine for their precious hints in my everyday work.
Finally, I am specially thankful to all friends and colleagues who have shared with me the joys and difficulties of such an exciting experience, particularly Vincenzo, Pietro and Marianna.

- CHAPTER 1 -

INTRODUCTION

§ 1.1 PROBLEM AND CHALLENGES

Automatic *indexing*, *retrieval* and *classification* of images are challenging tasks in computer vision that have attracted much research interest in the recent years. A huge number of different approaches have been proposed in order to clearly define such image understanding problems from a computational point of view and provide technical tools able to tackle them in an effective way. A broad class of such systems share a common paradigm, which splits the task of image indexing/classification in two main steps: (1) representation of the low-level visual content in a (high-dimensional) *feature space*, and (2) supervised learning of a *classification function* using annotated images and possibly a-priori knowledge. Several reference databases have been created in order to evaluate and compare different techniques, and the best performing methods on such data generally follow the mentioned paradigm. Some popular examples are represented by kernel machines (*e.g.*, support vector machines), boosting classifiers (*e.g., AdaBoost*) and prototype-based classifiers (*e.g.*, *k*-nearest neighbors), which have been repeatedly tailored to effectively deal with a number of image descriptors, ranging from local appearance descriptors to shape descriptors or global layout features. Although most of these techniques provide satisfactory results on standard image databases, their performances easily drop off when dealing with real search/classification problems, involving complex image collections with a huge number of images (billions and more) and categories. For instance, consider the problem of categorizing real-world scenes, where, according to a recent study of Xiao et al [XHE+10], one should consider almost one thousand categories that may overlap, due to the inherent hierarchical nature of concept-based annotation. Furthermore, even the best performing methods often cannot match the computa-

tion time constraints of these real applications. Therefore, designing effective methods that match the *scalability* requirement in terms of both size of image databases and number of tags/class labels is the main challenge of current research on image indexing/classification.

§ 1.2 Outline of the thesis

In this PhD thesis we address some major topics related to indexing and classification of images. In particular, we investigate the most relevant functional blocks of an image retrieval/categorization system, and propose original solutions to address some of their specific issues. First of all, we describe our original "Sparse Multiscale Patches" (SMP) image descriptor, which represents the texture information located at different resolution levels in a sparse way (Chap. 2). The SMP descriptor is associated to a new information-theoretic similarity measure, which can be computed in a k-nearest neighbor (k-NN) estimation framework. Then, in Chap. 3, we address the problem of how to efficiently organize data arising from images in order to perform fast k-NN search with respect to generic dissimilarity measures, like the broad class of Bregman divergences (symmetrized or not). The core part of the thesis is devoted to analyze and improve instance-based classification methods like the k-NN rule in a supervised framework that is inspired by boosting. We first describe our one-versus-all learning method, called "Universal Nearest Neighbors" (UNN), which brings the boosting principle into prototype-based classification, by defining prototypes as weak classifiers to be induced via the minimization of a risk functional (Chap. 4). Then, we describe our inherently multiclass MLNN algorithm (Chap. 5), that further generalizes the UNN approach by modifying the risk function in order to learn prototypes in all classes simultaneously, and allowing to use suitable density estimation kernels for improving the precision of local density classification. The main practical advantage of our method is to considerably reduce the size of the prototype set that is to be used at classification time, by rejecting "noisy" prototypes, thus enabling better precision at a reduced computational cost. We validated our novel approach on some well-known real-world image datasets, which contain from 8 to 15 challenging categories. Experiments show that the proposed method improves over the state-of-the-art prototype-based classification significantly, while comparing favourably with the best performing methods in terms of computational complexity. We also applied our method to the challenging problem of medical image classification, particularly focusing on the body part recognition from radiographic images (Chap. 6). Finally, we report a conclusion (Chap. 7) and two appendices that provide more mathematical details on our UNN (Appendix A) and MLNN approaches (Appendix B).

PART I

FEATURES AND SIMILARITY MEASURES FOR EFFECTIVE IMAGE INDEXING

- CHAPTER 2 -

SPARSE MULTISCALE PATCHES (SMP) FOR IMAGE RETRIEVAL

In this chapter we present a framework for defining an objective measure of the similarity (or dissimilarity) between two images for the purpose of image indexing. The problem is twofold:

- define a set of features that capture the information contained in the image in a relevant way for the given task;
- define a similarity measure in this feature space.

In particular, we propose a novel feature representation of images as well as a statistical information-theoretic measure on this representation. Our feature space is based on a global descriptor of images in a multiscale transformed domain. Indeed, after decomposition into a Laplacian pyramid, the transform coefficients are arranged in intra-scale/inter-scale/inter-channel vectors, that we call *patches*, reflecting the dependencies of neighboring coefficients in presence of specific structures or textures. At each scale, the probability density function (pdf) of these patches is used as a descriptor of the relevant information. Due to the sparsity of the multiscale transform, the most significant patches, called *Sparse Multiscale Patches (SMP)*, describe these pdfs efficiently. Thus we propose to use an information-theoretic distorsion measure (the Kullback-Leibler divergence) based on the comparison of these probability density functions. Interestingly, this measure can be effectively estimated via the *nonparametric*, k-nearest neighbor framework, without explicitly building the pdfs.

We tested our method on the query-by-example image retrieval task. Experiments on two publicly available databases showed the potential of our *SMP* approach for this task. In particular, our method provides performances comparable to those of a *SIFT*-based retrieval method and two

versions of a fuzzy segmentation-based method (the *UFM* and *CLUE* techniques), while showing good robustness to different geometric and radiometric deformations of the images.

§ 2.1 INTRODUCTION

2.1.1 Measuring similarities between images

Defining an objective measure of the similarity (or dissimilarity) between two images (or parts of them) is a challenging problem in many tasks of image processing and computer vision, which has been addressed from different perspectives. For instance, when dealing with inverse problems such as image denoising or deconvolution, a similarity measure is needed in order to assess how well the estimate explains the observations. However, for this class of problems, efforts have been concentrated in conditioning the inverse operator as well as the spatial properties of the estimated images. The measure of fitness to the data has been less studied and has been commonly defined as a simple Euclidean distance in the pixel space, such as: $d(I_1, I_2) = \sqrt{\sum_{i \in \{pixel\}} |I_1(i) - I_2(i)|^2}$. Conversely, when analyzing research in other applications, the similarity measure is at the core of the problem and has received much more attention. This is the case for tasks like tracking or image indexing/retrieval. Namely, in this latter application, the task consists in ranking the images in a database according to their visual similarity (or dissimilarity) to a given query image. Independently on the particular application, defining a similarity measure can be modeled as a two-steps process:

1. Define a set of properties (or features) that capture the visual information that is relevant for the given task. This step defines the so-called *feature space*.

2. Define a similarity measure between data (or subsets of data) in the feature space. This measure needs not to be necessarily a distance metric, as it should be adapted to the inherent geometric structure of the feature space.

A huge number of techniques have been investigated for solving the feature space problem. Some feature extraction approaches involve a transformed domain (*e.g.*, wavelet transforms), others are based on different descriptors of the appearance, shape or layout of images. A variety of descriptors (see [DKN08] for a review) has been proposed in the literature. Among these, *local descriptors* such as salient points of Loupias et al [LSBJ00] are based on a subset of pixels (typically arranged as blocks), whereas *global descriptors* give information about the image as a whole (*e.g.*,

color histograms [SB91]). Thus, on the one hand, local descriptors exploit the information given by a limited (generally variable) number of *interest points* along with their spatial neighborhood. The best known state-of-the-art local descriptor, widely used for image indexing/classification, is the SIFT descriptor [Low04], which has been shown to outperform the most common descriptors of interest regions [MS05]. On the other hand, global descriptors represent information of the whole image and range from the simplest histograms of gray-scale values or color histograms [SB91], to the more recent Bag-of-Feature histograms [SZ03], spatial pyramids of local descriptors [LSP06], Gist descriptors of the image layout [OT01], Fisher kernels [PD07] and VLAD descriptors [JDSP10]. Such descriptors are typically based on probabilistic models of the distribution of descriptor of local patches, thus not directly representing the visual information extracted at the pixel level. In our work, we concentrated on the so-called "global" patch descriptors, *i.e.*, global descriptor that include spatial and/or scale correlations at tha patch level, as supported by fundamental studies on image statistics [HM99].

So as for the dissimilarity measures, they range from simple metric distances like the Euclidean norm to more sophisticated distorsion measures. *E.g.*, robust estimators have been used for the optical flow [BA96], Bhattacharya distance for tracking [CRM00], entropic measures such as entropy, Kullback-Leibler divergence or mutual information for image registration [VI97, BSC+99]. A general requirement for the similarity measure is the visual relevance, *i.e.*, strong correlation with the human perception of similarity between images. Research in vision science has already brought some perspectives on how to accomplish such a requirement [MMBG06]. Nevertheless, designing systems merely based on the perceptual characteristics of the human visual system is a difficult task. Therefore, once meaningful features have been selected, metrics that are mathematically well-founded and can be easily implemented are generally preferred for applications. For this purpose, several distance metrics have been used to compare feature vectors for various tasks of image processing [PRTB99]. In this chapter we propose a new feature extraction method, as well as a statistical measure on the related feature space. In particular, we compute global descriptors in a transformed domain, that we call *Sparse Multiscale Patches*. Then, we propose to use a discrete estimation technique for computing an information-theoretic distorsion measure between the pdfs of such patches.

2.1.2 Proposed feature space and measure

We propose a new descriptor based on *Sparse Multiscale Patches*. In short, we propose to use probability distributions for integrating the local information brought by the *SMP*. The key aspects of these descriptors are the following:

- they provide a *multiscale* representation of the images;
- inter-/intra-scale patches describe the image structure locally at a given scale;
- they represent images in a *sparse* way, *i.e.*, most of the energy is concentrated in a few patches.

Note that the occurrence in different parts of an image of similar patches of spatially neighboring pixels has been already exploited in image processing [BCM05, AW06, ADB08]. In our work, we ground on the same idea generalized to multiscale coefficients, as proposed by Portilla et al [PSWS03].

The visual content of images is represented by patches of multi-resolution coefficients. The extracted feature vectors are viewed as samples from an unknown multi-dimensional distribution. The multi-resolution transform of an image being sparse, a reduced number of patches yields accurate approximation of the underlying distribution. Then, we estimate the similarity between images by a pseudo-distance (or distorsion measure) between these multi-dimensional probability density functions. Namely, we propose to use the Kullback-Leibler (KL) divergence as a measure for quantifying the dissimilarity between two probability density functions. This measure has already shown good performances in the context of image retrieval [PRTB99]. It has been already used for the simple case of parameterized marginal distributions of wavelet coefficients [DV02, WWS+06], assuming independence of the coefficients. In contrast, we define multi-dimensional feature vectors (patches), that capture *inter-scale* and *intra-scale* dependencies among subband coefficients. These are better adapted to the descriptor of local image structures and texture. In addition, for color images, we take into account the dependencies among the three color channels; hence patches of coefficients are also *inter-channel*. This approach should involve estimating distributions in a high-dimensional statistical space, where fixed-size kernel options to estimate distributions or divergences fail. Alternatively, we propose to estimate the KL divergence directly from the samples by using the k-nearest neighbor (k-NN) approach, *i.e.* adapting to the local sample density.

In this chapter, we first define our feature space, which relies on inter-/intra-scale and inter-channel patches of Laplacian pyramid coefficients for color images, called *Sparse Multiscale Patches* (Sec. 2.2). Then we define the global similarity on this feature space by combining similarities between the probability density functions of these patches at different scales (Sec. 2.3). The comparison between pdfs is measured by the KL divergence. We also explain how to estimate this quantity. Finally, in the last section we illustrate the use of the proposed measure for the image retrieval task.

§ 2.2 Feature space: Sparse Multiscale Patches

Throughout this chapter, we will denote the input image by I, the scale of the multi-resolution decomposition by j, and the location in the 2D image space by k.

2.2.1 Multiscale coefficients: advantages and drawbacks

The wavelet transform enjoys several properties that have made it very successful in image processing and that are relevant for the definition of similarity between images. Indeed, it provides a sparse representation of images, meaning that it concentrates the information content of an image into few coefficients of large magnitude while the rest of the coefficients are small. This aspect, combined with a fast transform, makes wavelet thresholding methods very powerful. Indeed, only selecting large coefficients is sufficient for discriminating where the most relevant information is localized in the image. For example, denoising can be done very efficiently by simply thresholding wavelet coefficients, as proved in [DJ94]. Such simple coefficient-wise treatments provide results of excellent quality at a very reduced computational cost.

In fact, these classical wavelet methods treat each coefficient separately, relying on the fact that they are decorrelated. However, the wavelet coefficients are not independent and these dependencies are the signature of structures present in the image. For example, a discontinuity between smooth regions at point k_0 will give large coefficients at this point at all scales j ($w(\mathrm{I})_{j,k_0}$ large for all j). Classical methods using coefficient-wise treatments may destroy these dependencies between coefficients and hence alter the local structure of images. Therefore, models exploiting dependencies between the coefficients have been proposed and successfully used for image enhancement (e.g. [PSWS03, RCB01]). In particular, Portilla et al [PSWS03] introduced the concept of patches of wavelet coefficients (which they called "neighborhoods of wavelet coefficients") in order to represent efficiently fine spatial structures in images.

2.2.2 Multiscale patches for color images

Following these ideas, we define a feature space based on a sparse description of the image content by means of a multi-resolution decomposition. More precisely, in order to build a multi-resolution version of an image, we propose to use the Laplacian pyramid [BA87]. Originally proposed for image compression, this technique consists in applying a Gaussian pyramid, *i.e.*, multiresolution filtering using a Gaussian filter, and then retaining at each level the "error" image, that is the difference between images at two

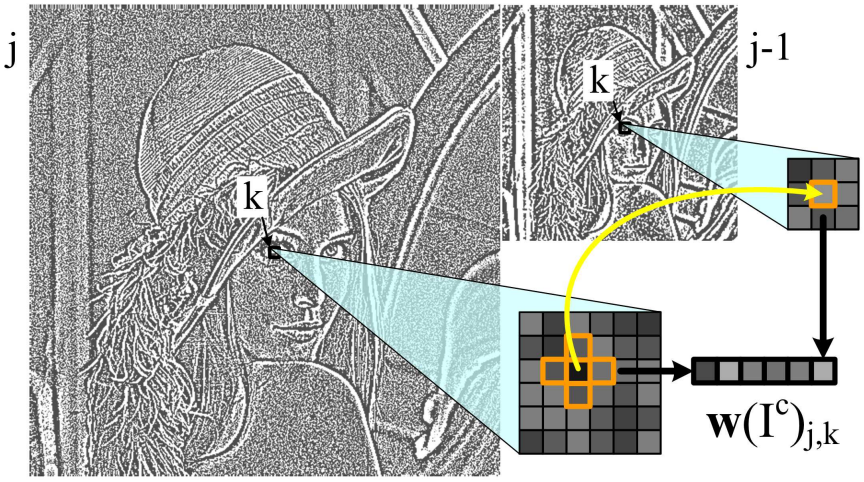


Figure 2.1: *Building a patch of multiscale coefficients, for a single color channel image.*

consecutive levels of the Gaussian pyramid. In order to build our descriptor, we group the Laplacian pyramid coefficients of the three color channels of image I into coherent vectors called patches. Here the coherence is sought by grouping coefficients linked to a particular scale j and location k in the image.

In fact, the most significant dependencies are seen between a coefficient $w(\mathrm{I})_{j,k}$ and its closest neighbors in space: $w(\mathrm{I})_{j,k\pm(0,1)}$, $w(\mathrm{I})_{j,k\pm(1,0)}$ and in scale: $w(\mathrm{I})_{j-1,k}$, where $j-1$ represents the scale a step coarser than the scale j. Grouping the closest neighbors in scale and space of the coefficient $w(\mathrm{I})_{j,k}$ in a vector, we obtain the patch $\vec{w}(\mathrm{I})_{j,k}$ (see Fig. 2.1):

$$\vec{w}(\mathrm{I})_{j,k} = \left(w(\mathrm{I})_{j,k}, w(\mathrm{I})_{j,k\pm(1,0)}, w(\mathrm{I})_{j,k\pm(0,1)}, w(\mathrm{I})_{j-1,k}\right) \tag{2.1}$$

which describes the structure of the grayscale image I at scale j and location k. The probability density functions of such patches at each scale j have proved to characterize fine spatial structures in grayscale images [PSWS03, PAHD05]. Such patches are therefore relevant features for our problem as will be seen in Section 2.4.3.

We consider color images in the luminance/chrominance space: $\mathrm{I} = (\mathrm{I}^Y, \mathrm{I}^U, \mathrm{I}^V)$. Since the coefficients are correlated through channels, we aggregate in the patch the coefficients of the three channels:

$$\mathbf{w}(\mathrm{I})_{j,k} = \left(\vec{w}(\mathrm{I}^Y)_{j,k}, \vec{w}(\mathrm{I}^U)_{j,k}, \vec{w}(\mathrm{I}^V)_{j,k}\right) \tag{2.2}$$

with $\vec{w}(\mathrm{I}^Y)_{j,k}$, $\vec{w}(\mathrm{I}^U)_{j,k}$ and $\vec{w}(\mathrm{I}^V)_{j,k}$ given by Eq. (2.1).

The low-frequency approximation resulting from the Laplacian pyramid is also used fro building additional feature vectors. Namely, 3×3 pixel

neighborhoods along all three channels are joined together to form patches of dimension 27 (whereas patches from the higher-frequency subbands are of dimension 18, as defined in Eq. (2.2)). The union of the higher-frequency and low-frequency patches forms our feature space. The patches of this augmented feature space will be denoted by $\mathbf{w}(\mathrm{I})_{j,k}$.

2.2.3 Multiscale transform

The coefficients are obtained by a Laplacian pyramid decomposition [BA87]. Indeed, critically sampled tensor wavelet transforms lack rotation and translation invariance and so would the patches made of such coefficients. Thus we prefer to use the Laplacian pyramid which shares the sparsity and inter-/intra-scale dependency properties with the wavelet transform while being more robust to rotations.

2.2.4 Sparsity of the multiscale patches

As explained above, multiscale coefficients provide a sparse representation of images by concentrating the information into few coefficients of large amplitude and this sparsity is exploited by thresholding methods on the raw coefficients. As illustrated in Fig. 2.2, our experiments show that the patches of multiscale coefficients of large overall energy (sum of the square of all coefficients in a patch) also concentrate the information. More precisely, we selected a fixed proportion of patches at each scale of the decomposition and proved that the resulting similarity measure (defined in Sec. 2.3) remains consistent. Since the total number of patches is $4/3N$ with N the number of pixels in an image, the number of samples we have in the feature space is quite large as far as measuring a similarity is concerned. The possibility of selecting a small number of patches which represent the whole set well is highly desirable and we will exploit it to speed up our computations.

Note that other selecting procedures may be investigated such as using the energy of the central coefficient, using the sum of absolute differences in the patches or thresholding based on the variance of the patches.

Let us now explain how we define a similarity in this feature space.

§ 2.3 SIMILARITY MEASURE

2.3.1 Definition

Our goal is to define a similarity measure between two images I_1 and I_2 from their feature space i.e. from their respective set of patches $\{\mathbf{w}(\mathrm{I}_1)_{j,k}\}_{j,k}$ and $\{\mathbf{w}(\mathrm{I}_2)_{j,k}\}_{j,k}$. When images are clearly similar (e.g. different views of the same scene, images containing similar objects...), their patches $\mathbf{w}(\mathrm{I}_1)_{j_o,k_o}$

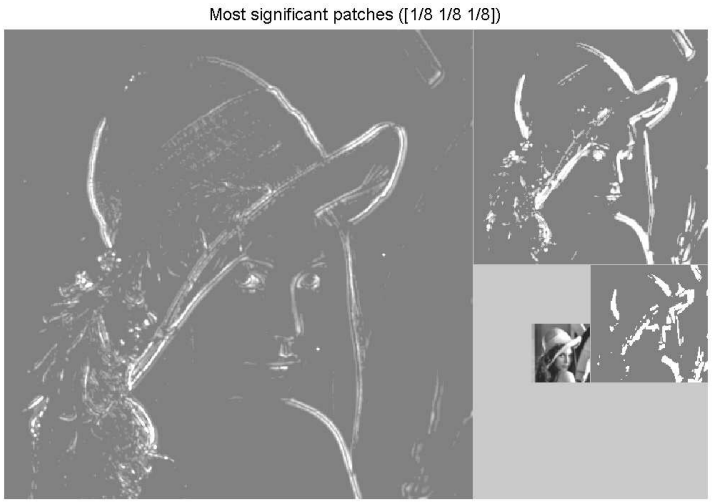


Figure 2.2: *White indicates the location of patches of largest energy (1/8 of the patches is selected for each subband).*

and $\mathbf{w}(\mathrm{I}_2)_{j_o,k_o}$ do not necessarily correspond. Hence a measure comparing geometrically corresponding patches would not be robust to geometric transformations. Thus, we propose to compare the pdfs of these patches. Specifically, for an image I, we consider for each scale j the pdf $p_j(\mathrm{I})$ of the set of patches $\{\mathbf{w}(\mathrm{I})_{j,k}\}_k$.

In order to compare two pdfs, we use the Kullback-Leibler divergence, that is a Bregman divergence defined from the Shannon differential entropy as generator function (see Chap. 3), and quantifies the amount of information in a random variable through its pdf. Namely, the Kullback-Leibler divergence (D_{kl}) is defined as follows [PRTB99]:

$$D_{kl}(p_1||p_2) = \int p_1 \log(p_1/p_2), \tag{2.3}$$

This divergence has been successfully used for other applications in image processing in the pixel domain [BDB07, ADB08], as well as for evaluating the similarity between images using the marginal pdf of the wavelet coefficients [DV02, WWS+06]. In this chapter, we propose to measure the similarity $S(\mathrm{I}_1, \mathrm{I}_2)$ between two images I_1 and I_2 by summing over scales the divergences between the pdfs $p_j(\mathrm{I}_1)$ and $p_j(\mathrm{I}_2)$:

$$S(\mathrm{I}_1, \mathrm{I}_2) = \sum_j \alpha_j D_{kl}(p_j(\mathrm{I}_1)||p_j(\mathrm{I}_2)) \tag{2.4}$$

where α_j is a positive weight that may normalize the contribution of the different scales.

2.3.2 Limits of the parametric approaches to the estimation

The estimation of the similarity measure S consists of the evaluation of divergences between pdfs $p_j(\mathrm{I_i})$ of high dimension. This raises two problems. First, estimating the KL divergence, even with a good estimate of the pdfs, is hard because this is an integral in high dimension involving unstable logarithm terms. Secondly, the accurate estimation of a pdf itself is difficult due to the lack of samples in high dimension (curse of dimensionality). The two problems should be embraced together to avoid cumulating both kinds of errors.

A first idea consists in parametrizing the shape of the pdf. The marginal pdf of multiscale coefficients is well modeled by generalized Gaussians. In this case, the KL divergence is an analytic function of the pdf parameters. This technique has been used in [DV02, WWS+06] to compare images on the basis of the marginal pdf of their wavelet coefficients. To our knowledge, the generalized Gaussian model cannot be extended to account for correlations in higher dimension. Mixture of Gaussians on the other hand are efficient multi-dimensional models accounting for correlations that fit well the pdf of wavelet coefficients patches [PAHD05]. However the KL divergence is not an analytic function of the model parameters.

Thus, we propose to make no hypothesis on the pdf at hand. We therefore spare the cost of fitting model parameters but we have to estimate the divergences in this non-parametric context. Conceptually, we combine the Ahmad-Lin approximation of the entropies necessary to compute the divergences with "balloon estimates" of the pdfs using the k-NN approach.

2.3.3 Non-parametric estimation of the similarity measure

The KL divergence can be written as the difference between a cross-entropy $\mathrm{H_x}$ and an entropy H (see Eq. (2.3)):

$$\mathrm{H_x}(p_1, p_2) = -\int p_1 \log p_2, \quad \mathrm{H}(p_1) = -\int p_1 \log p_1. \tag{2.5}$$

Let us explain how to estimate these terms from an i.i.d sample set $\mathcal{W}^1 = \{\mathbf{w}_1^1, \mathbf{w}_2^1, .., \mathbf{w}_{N_1}^1\}$ of p_1 and an i.i.d sample set $\mathcal{W}^2 = \{\mathbf{w}_1^2, \mathbf{w}_2^2, .., \mathbf{w}_{N_2}^2\}$ of p_2. (The samples are in $\mathbb{R}^d$.)

Assuming we have estimates $\widehat{p_1}$, $\widehat{p_2}$ of the pdfs p_1, p_2, we use the Ahmad-Lin entropy estimators [AL76]:

$$\mathrm{H_x^{al}}(\widehat{p_1}, \widehat{p_2}) = -\tfrac{1}{N_1} \sum_{n=1}^{N_1} \log[\widehat{p_2}(\mathbf{w}_n^1)], \quad \mathrm{H^{al}}(\widehat{p_1}) = -\tfrac{1}{N_1} \sum_{n=1}^{N_1} \log[\widehat{p_1}(\mathbf{w}_n^1)]. \tag{2.6}$$

General non-parametric pdf estimators from samples can be written as a sum of kernels K with (possibly varying) bandwidth h (see [TS92] for a

review):

$$\widehat{p_1}(x) = \frac{1}{N_1} \sum_{n=1}^{N_1} K_{h(\mathcal{W}^1,x)}(x - \mathbf{w}_n^1). \tag{2.7}$$

- Parzen estimators $h(\mathcal{W}^1, x) = h$: the bandwidth is constant. They perform very well with samples in one dimension but become unadapted in high dimension due to the sparsity of the samples: the trade-off between a bandwidth large enough to perform well in low local sample density (which may *oversmooth* the estimator) and a bandwidth small enough to preserve local statistical variability (which may result in an unstable estimator) cannot always be achieved. To cope with this problem, kernel estimators using adaptive bandwidth have been proposed;

- Sample point estimators $h(\mathcal{W}^1, x) = h_{\mathcal{W}^1}(\mathbf{w}_i^1), i \in \{1, N_1\}$: the bandwidth adapts to each sample $\mathbf{w}_i^1$ given the sample set $\mathcal{W}^1$;

- Balloon estimators $h(\mathcal{W}^1, x) = h_{\mathcal{W}^1}(x)$: the bandwidth adapts to the point of estimation x given the sample set $\mathcal{W}^1$.

We use a balloon estimator with a binary kernel and a bandwidth computed in the k-nearest neighbor (k-NN) framework [TS92]. This is a dual approach to the fixed-size kernel methods and was firstly proposed by Loftsgaarden and Quesenberry [LQ65]: the bandwidth adapts to the local sample density by letting the kernel contain exactly k neighbors of x among a given sample set:

$$K_{h(x)}(x - \mathbf{w}_n^1) = \frac{1}{v_d \, \rho_{k,\mathcal{W}^1}^d(x)} \, \delta\Big[||x - \mathbf{w}_n^1|| < \rho_{k,\mathcal{W}^1}(x)\Big] \tag{2.8}$$

with v_d the volume of the unit sphere in $\mathbb{R}^d$ and $\rho_{k,\mathcal{W}}(x)$ the distance of x to its k-th nearest neighbor in $\mathcal{W}$. Although this is a biased pdf estimator (it does not integrate to one), it has proved to be efficient for high-dimensional data [TS92]. Plugging Eq. (2.8) in Eq. (2.6), we obtain the following estimators of the (cross-)entropy:

$$\mathrm{H}^{\mathrm{knn}}(\widehat{p_1}) = \log[N_1 \, v_d] - \log(k) + \frac{d}{N_1} \sum_{n=1}^{N_1} \big(\log[\rho_{k,\mathcal{W}^1}(\mathbf{w}_n^1)]\big) \ , \tag{2.9}$$

$$\mathrm{H}_{\mathrm{x}}^{\mathrm{knn}}(\widehat{p_1}, \widehat{p_2}) = \log[N_2 \, v_d] - \log(k) + \frac{d}{N_1} \sum_{n=1}^{N_1} \big(\log[\rho_{k,\mathcal{W}^2}(\mathbf{w}_n^1)]\big) \ . \tag{2.10}$$

As previously stated, these estimates are biased. A correction of the bias has been derived in [GLMI05] in a different context. In the non-biased

estimators of the (cross)-entropy the digamma function $\psi(k)$ replaces the $\log(k)$ term:

$$\mathrm{H}^{\mathrm{knn}}(\widehat{p_1}) = \log[(N_1-1)v_d] - \psi(k) + \frac{d}{N_1}\sum_{n=1}^{N_1}\left(\log[\rho_{k,\mathcal{W}^1}(\mathbf{w}_n^1)]\right) \ , \quad (2.11)$$

$$\mathrm{H}_{\times}^{\mathrm{knn}}(\widehat{p_1},\widehat{p_2}) = \log[N_2\ v_d] - \psi(k) + \frac{d}{N_1}\sum_{n=1}^{N_1}\left(\log[\rho_{k,\mathcal{W}^2}(\mathbf{w}_n^1)]\right) \ . \quad (2.12)$$

And hence the KL divergence reads:

$$D_{kl}(p_1||p_2) = \log\left[\tfrac{N_2}{N_1-1}\right] + \tfrac{d}{N_1}\sum_{n=1}^{N_1}\log[\rho_{k,\mathcal{W}^2}(\mathbf{w}_n^1)] - \tfrac{d}{N_1}\sum_{n=1}^{N_1}\log[\rho_{k,\mathcal{W}^1}(\mathbf{w}_n^1)] \ . \quad (2.13)$$

This estimator is valid in any dimension d and robust to the choice of k.

§ 2.4 Application: Image Retrieval

2.4.1 Content-based image retrieval

With the rapid growing of general-purpose image collections, performing efficiently a search on such large datasets becomes a more and more critical task. Content-based image retrieval (CBIR) systems tackle this task by analyzing the content of images in order to provide meaningful signatures of them. Automatic search of the target images is made possible by defining a similarity measure on the underlying signature space which has a reduced dimension. As a result, content based image retrieval mainly relies on describing the image content in a relevant way (the feature space) and defining a quantitative measure on this space (the similarity measure): the retrieval task is then accomplished by ranking images in increasing order of the pseudo-distance between their feature vector and the one of a given query image.

As seen in the introduction, a variety of descriptors and similarity measures have been proposed to represent image content. In this chapter, we will compare our *SMP* approach to three different approaches to image retrieval, two of them relying on the same similarity measure. The first approach is based on *SIFT* descriptors [Low04], which are considered state-of-the-art amongst local descriptors. Salient points are extracted by detecting the highest coefficients in the wavelet transform of the image and *SIFT* features are then represented by histograms of the gradient orientation in regions of interest. Matching the *SIFT* features obtained in two images allows then to quantify their similarity. The other methods to which we compared ours use a segmentation-based fuzzy logic approach called *UFM* for *Unified Feature Matching* [CW02]. Here the descriptors are fuzzy features

(called fuzzy sets) reflecting the color, texture, and shape of each segmented region. The *UFM* measure then integrates the fuzzy properties of all the regions to quantify the similarity between two images. Using this measure, the authors proposed two image retrieval algorithms. The first one is a strictly content-based approach (similarly to ours): it consists in ranking the database images based solely on their *UFM* distance to the query. We refer to it as the *UFM* approach. The retrieval accuracy is improved by a second method called CLUE: the *UFM* distances between target images themselves are used to obtain a clustering of the data from which the ranking is obtained. Consequently, this method involves additional information compared to strict content-based systems such as our approach.

2.4.2 Database and parameter settings

2.4.2.1 Databases

Numerical experiments were performed on two different databases. The first one contains small categories and allows to evaluate specific performances of a retrieval system such as its robustness to deformations; while the second database, with larger categories, allows to test global retrieval performances.

One of these databases contains 1,000 images of the Nister Recognition Benchmark collection [NS06]. The images of size 640x480 pixels are grouped by sets of four images showing the same scene or object. Their content is quite various, from indoor scenes with a single object to outdoor scenes. Images belonging to the same group are related by geometric deformations (rotation, translation, zoom and perspective) as well as radiometric deformations (changes of brightness and contrast). The ground-truth for any query image is clear: exactly the three other images of the same group are relevant.

The *SMP* retrieval method was also tested on a general-purpose image database from COREL that has been widely used for CBIR evaluation purposes. In particular, results presented in [CWK05] can be considered as a reference. We used the same subset of the COREL database as in [CWK05]. It includes 1,000 images of size 384×256 or 256×384 which are classified in 10 semantic categories (*Africa, Beach, Buildings, Buses, Dinosaurs, Flowers, Elephants, Horses, Food, Mountains*). In some categories, the visual similarity between two given images is not always obvious since the grouping has been made in a semantic sense (e.g., category "Africa").

2.4.2.2 Parameter settings

To build the patches as defined in section 2.2.2, the Laplacian pyramid was computed for each channel of the image (in the YUV color space) with a

5-point binomial filter $w_5 = [1\ 4\ 6\ 4\ 1]/16$, which is a computationally efficient approximation of the Gaussian filter classically used to build Laplacian pyramids. Three high-frequency subbands plus the low-frequency approximation were used.

In the following experiments, $1/64$ (resp. $1/32$, $1/16$ and all) of the patches were selected in the first high-frequency (resp. second, third and low-frequency) subband to describe an image (see 2.2.4). At each scale, the KL divergence was estimated in the kNN framework, with $k = 10$. The contributions to the similarity measure from the divergences of all subbands were equally weighted ($\alpha_j = 1$ in Eq. (2.4)).

Note that the use of the Jensen-Shannon divergence, which is a symmetrized version of the KL divergence, has also been studied. We found that the performances of this symmetric measure are less good than with the KL divergence, and so until further understanding of this phenomenum, we report here only the results with the KL divergence.

2.4.3 Numerical experiments

This section presents an experimental analysis of the *SMP* method; the patch-based retrieval algorithm is evaluated in terms of its ability to retrieve similar images in a query-by-example context. Images belonging to the Nister database were used to evaluate the robustness of the method to different geometric transformations. A set of artificially-degraded images of this database was also used to evaluate the retrieval performances with respect to radiometric deformations (JPEG2000 compression noise). The global retrieval performances on the Nister database were evaluated by ROC (Receiver Operating Characteristic) curves and our method was compared to a reference *SIFT*-based retrieval algorithm. For the COREL database, the global retrieval performances were evaluated by precision curves and our method was compared with the fuzzy, segmentation-based *UFM* approach. Note that for all the following experiments, the given distance between images is S (Eq. (2.4)), hence the smaller the given distance is the more similar the two considered images are.

2.4.3.1 Robustness to geometric deformations

The robustness of a retrieval system to geometric deformations is its ability to find relevant images in spite of some transformations of the query, such as changes of viewpoint, rotations, zoom. This is an important requirement in image retrieval, e.g. for finding a given object in different images independently of the viewpoint. Because of its structure, the Nister database allows to evaluate the robustness of the proposed method to geometric deformations. Indeed, the database is composed of groups of four images containing the same object or scene under different viewpoints

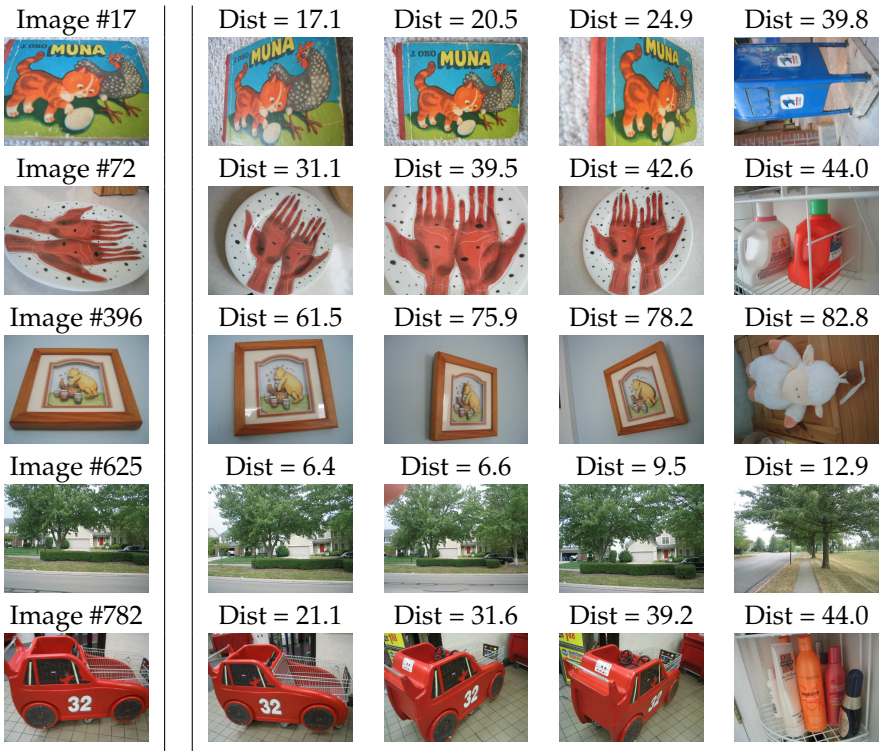


Figure 2.3: *Retrieval results for 5 images of the Nister database. For each row, left to right: query image; first 4 ranked images of the database (excluding the query). For each retrieved image, the distance to the query is also shown (smaller distances meaning more similar).*

and/or lightening conditions.

Examples of retrieval for five query images taken from the database are presented in Fig. 2.3. In this figure, each row displays the retrieval result for the query image shown in the leftmost column. From the second column on, one can see the first 4 retrieved images ranked in increasing order of their distance to the query. Hence the second leftmost image is the most similar one, excluding the query image which is always ranked first with a distance of zero. The first retrieved images are generally relevant for the query, in spite of rotations (row 2), changes of viewpoint (rows 1, 3, 5) and zooms (rows 2, 4). This shows that the proposed descriptors and similarity measure are robust in terms of geometric deformations for the retrieval problem.

2.4.3.2 Robustness to JPEG2000 compression

Another important requirement for content-based retrieval systems is the robustness to radiometric deformations. Transmission on heterogeneous

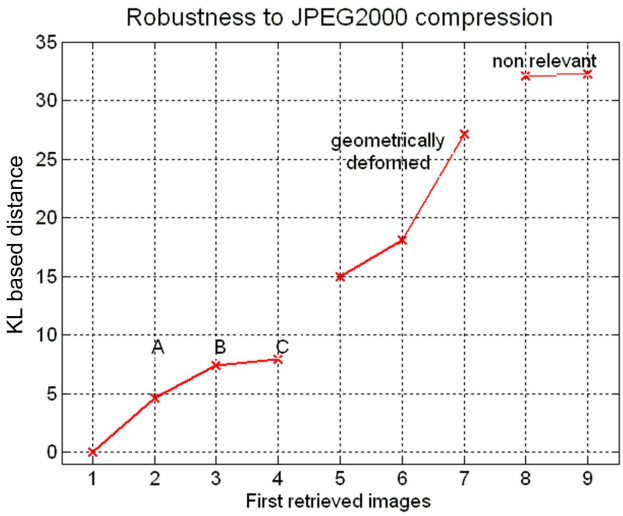


Figure 2.4: *Evaluation of the robustness to JPEG compression for one query image. Displayed distances are from the query to the 6 relevant images - 3 compressed (A, B, C) and 3 geometrically transformed versions of the query - and to the first 2 non-relevant images. PSNR of the compressed versions: A:* 31.8*dB, B:* 29.7*dB and C:* 29.3*dB.*

networks requires compression. This process induces a loss of quality that can be significant especially in critical transmission conditions. A retrieval system is expected to be robust to compression quality. To test the proposed method on this specific point, groups of images from the Nister database were expanded. Namely, three highly-compressed versions of one image were added to each group. They were obtained by setting three different quality levels of JPEG2000 compression.

Queries were launched on this dataset with both original and compressed images. An example of the results is shown in Fig. 2.4, where a non-compressed image is used as a query. The distance from the three compressed versions to the query image being quite small, the system ranked them first and before any geometrically deformed version of the query. This behavior is general and still holds when compressed images are used as queries, confirming the reliability of the proposed similarity measure in terms of its robustness to compression. Moreover, the distance to the query increases as the compression level increases. This is shown in Fig. 2.4, where images A, B, C are compressed versions of the query image in decreasing order of quality, the PSNR being respectively of 31.8, 29.7 and 29.3 dB.

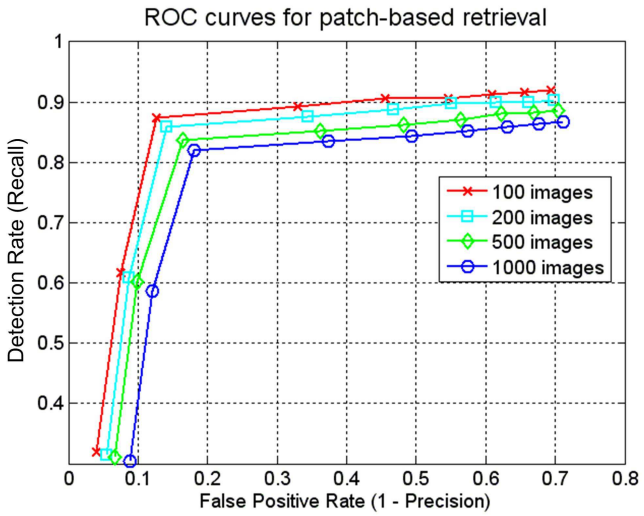


Figure 2.5: *Retrieval performance of the* SMP *method for different subset sizes of the Nister database; the* ROC *curves were obtained for cut-off values ranging from 1 to 9.*

2.4.3.3 Image retrieval performances (I): ROC curves and comparison with a *SIFT*-based method

The overall performances of the *SMP* retrieval method were evaluated by analyzing retrieval results on the Nister dataset; namely, each of the 1,000 images was used as a query and the similarity measure to all other images was computed. The same experiment was conducted by using a state-of-the-art retrieval method based on (local) *SIFT* descriptors [Low04]. For this method, the similarity measure is defined as the number of points of interest that can be matched between two images. The results of both methods were quantitatively compared by means of ROC curves. These are *recall* versus $1 - precision$ curves[1] averaged over all queries. The larger the precision and recall values, the better the retrieval performances (this corresponds to the top left side of the plot of an ROC curve).

The results of our *SMP* retrieval method are shown in Fig. 2.5 for different subset sizes of the database. Namely, average results on the first 100, 200 or 500 images are compared to those on the whole dataset (1000 images). Although the probability of retrieval errors increases with the size of the database, global performance is still satisfactory for a larger dataset. In any case, the best trade-off between precision and recall was reached when we retrieved three images, i.e. when the cut-off value matches exactly the

[1] *Recall* or *positive rate* $= \frac{D}{R}$, *1-precision* or *false positive rate* $= 1 - \frac{D}{C}$, with *R=#{relevant images for a given query}*, *C=#{desired number of retrieved images}* or *cut-off*, *D=#{correctly detected images}*.

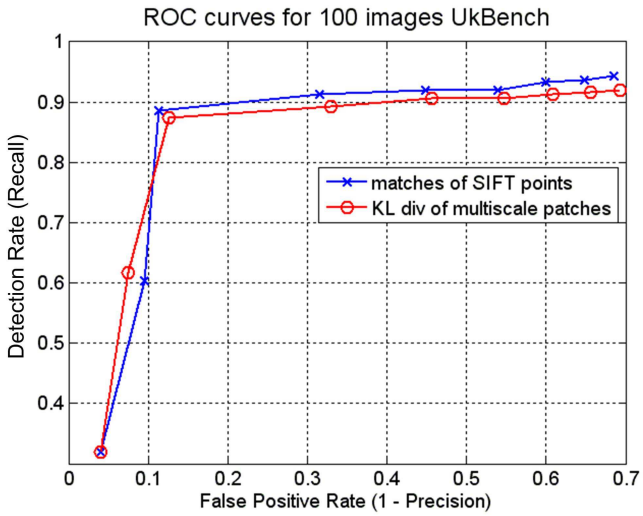


Figure 2.6: *Comparison of the retrieval performances of the* SMP *approach and the* SIFT*-based algorithm; the* ROC *curves were obtained for cut-off values going from 1 to 9.*

number of relevant images; as a result, there is a high probability that the retrieved images are all and only the relevant ones.

Finally, the results for our *SMP* and the *SIFT*-based approach are shown in Fig. 2.6. The latter were obtained by running a publicly available Matlab implementation of the *SIFT* algorithm [Low04]. Because of the long processing time of the SIFT implementation (4.8 s on average for each comparison between two images), performing a query with each image of the database could not be done in a reasonable time. In consequence, a comparison was made by querying a subset of 100 images. In light of the ROC curves, the performances of our *SMP* method and the *SIFT*-based algorithm are comparable for this experiment.

2.4.3.4 Image retrieval performances (II): precision curve and comparison with the *UFM* method

The *SMP* retrieval method was also tested on a subset of the COREL database and compared to the *UFM* and *CLUE* methods [CWK05]. This database is made of a small number of categories (10) containing a large number of images per category (100). Hence, ROC curves are not adapted to evaluate the global performances of a retrieval system in this case. Instead, we used the *Average Precision* to evaluate the retrieval performances for each category (the precision values for a cut-off equal to 100 were averaged over all images of the category) as in [CWK05].

Examples of our retrieval results are shown in Fig. 2.7 and the *Aver-*

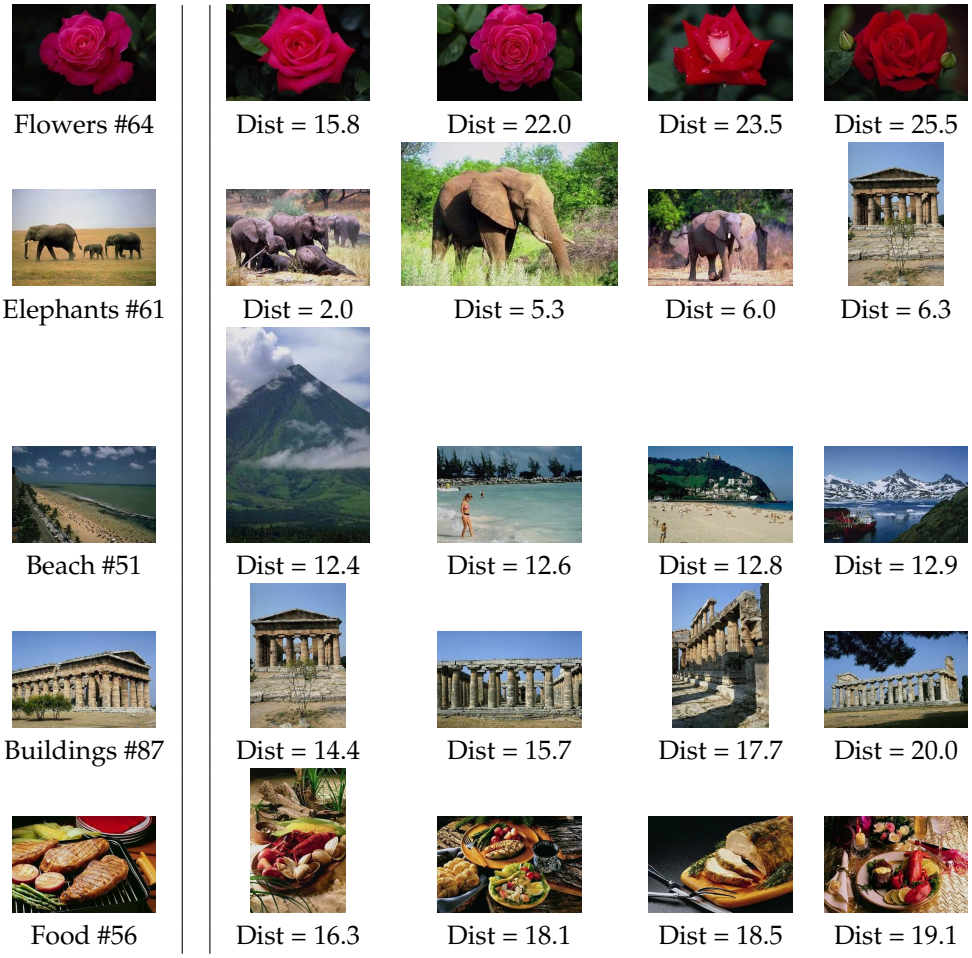


Figure 2.7: *Retrieval results for 5 images of the* COREL *database. For each row, left to right: query image; first 4 ranked images of the database (excluding the query image). For each retrieved image, the* SMP *similarity measure to the query is also shown.*

age Precision is given for each category in Fig. 2.8 (dark blue bars). The results of the *UFM* and *CLUE* approaches are also displayed in this latter figure for comparison. Fig. 2.7 illustrates the fact that the most of time, the first four retrieved images belong to the query's category (row 1, 4, and 5). This figure also illustrates well the difficulties encountered in this task: since the categories are quite large and diverse, images belonging to different categories may have very similar visual properties that are picked by our method. For example, the elephant and building (row 2 of Fig. 2.7) have dominating vertical structures and same dominant colors. Likewise, images belonging the "mountains" or "beaches" are freqently mismatched (row 3 of Fig. 2.7). These retrieval errors are common to all methods comparing images solely on the basis of the image content (i.e. introducing no semantics) and explain the fluctuation of the results displayed in Fig. 2.8 for all three methods. Our method compares well with the two established methods displayed here: it is more accurate than *UFM* (gray bars) for six categories out of ten; the accuracy is also better than or comparable to *CLUE* (white bars) for five categories out of ten. On average, our method performs better than the *UFM* approach and slightly less well than the *CLUE* one. As pointed out in Section 2.4.1, the *SMP* and *UFM* approaches are strictly content-based approaches. The *CLUE* method, while performing better, uses additional image distances and is therefore much more time-consuming. Thus, the performances of our method seem quite promising for three reasons:

- It performs slightly better than the *UFM* approach which relies on the same information.
- The results are not far from those of the more advanced *CLUE* approach which relies on more information.
- A similar clustering processing as the one applied with the *UFM* measure in *CLUE* may be applied to improve the *SMP* approach.

In conclusion, in its current state of development, the proposed *SMP* measure does not outperform the state-of-the-art methods selected as benchmark here. However, it does bring a novel approach to tackle the problem of image retrieval.

2.4.4 Computational speed-up(s)

The evaluation of our *SMP* similarity requires the computation of several KL divergences in a non-parametric framework. Since this is a time-consuming task, we propose two ways to speed-up the computations. The first one is based on a GPU implementation of the algorithm, the second on a preselection of the relevant images in the database.

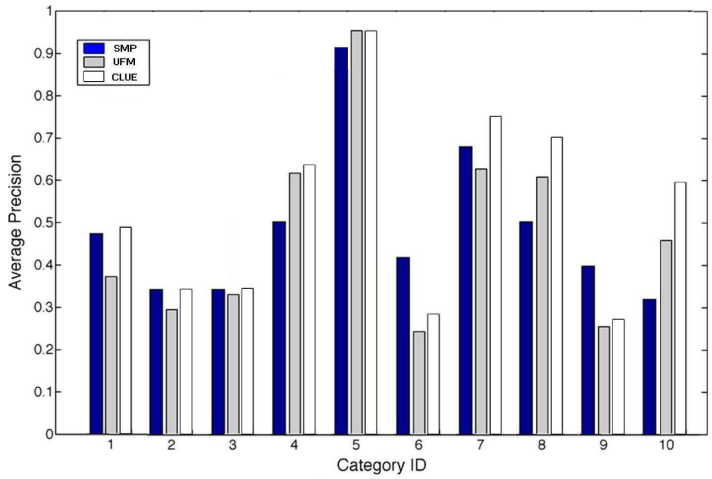


Figure 2.8: *Average Precision for each category of the* COREL *database. Dark blue bars:* SMP *approach; gray bars:* UFM *approach; white bars:* CLUE *approach.*

2.4.4.1 GPU implementation

When computing the similarity between two images with the *SMP* approach, most of the time is devoted to the search of the k-th nearest neighbors in the evaluation of the KL divergences. Indeed, finding a k-th nearest neighbor requires to compute and sort distances between features (here the patches). The "brute force" algorithm has a complexity of order $O(N^2)$ for N samples in the feature set. Smarter algorithms with a lower complexity (typically of order $O(N \log N)$) such as the classical KD-tree (ANN) algorithm [AMN+98] have been designed. Nevertheless, in practice, the computation time of a similarity between two images with the *SMP* approach remains large even with this low-complexity algorithm: on average 2.2s on a Pentium 4 3.4 GHz (2GB of DDR memory) with the ANN algorithm.

To speed up the computation time, we developed a parallel implementation of the kNN search on a Graphic Processing Unit (GPU) [GDB08] using CUDA. This implementation is based on a brute force approach since recursive algorithms (the preferred strategy when using trees such as in ANN) are not parallelizable. It was implemented on an NVIDIA GeForce 8800 GTX card with 768 MB of memory. The computation time for one similarity measure between two images required 0.2s on average (i.e., 10 times less than with the CPU implementation of ANN).

As of today, the brute force algorithm parallelized on GPU is by far the fastest implementation of our method. Developing smart algorithms (such as the KD-tree one), which may not be parallelizable but have a very low complexity, is a topic of active research, as is the development of GPU for computational purposes. Hence both types of methods should be kept in mind for efficient implementations in the near future.

2.4.4.2 Preselection of the relevant images

The computational speed can be improved by splitting the retrieval task into two steps:

1. Only the low frequency contribution to the similarity measure defined in Eq. (2.4) is computed for all images in the database. This "partial" similarity measure produces a first ranking of the database images from which the first n images are selected for the next step.

2. The complete similarity measure is computed between the query and the n selected images.

This procedure saves computation time as it computes the whole similarity measure only for a reduced number of images (computing only part of it for images that are unlikely to be relevant to the query). The smaller the size of the preselected subset, the greater the improvement in terms of computation time. For example, when a query on the Nister database is processed following the described two-step procedure with a selected subset of 50 images, the average computation time per image drops from 0.2s to about 0.06s with the GPU implementation (and with similar retrieval performances). It is clear however that the number of preselected images cannot be arbitrarily small without seriously affecting retrieval performances. It should be large enough compared to the number of images in the database as well as the number of relevant images for the query.

§ 2.5 CONCLUSION

In this chapter, we proposed a new image similarity framework based on high-dimensional probability distributions of patches of multiscale coefficients which we call *Sparse Multiscale Patches*. Feature sets are represented by these patches of subband coefficients that take into account intrascale, interscale and interchannel dependencies. The similarity between two images was defined as a linear combination of the "closeness" between the distributions of their features at each scale measured by the KL divergence. The KL divergences are estimated in a non-parametric framework, via a k-NN approach. The proposed similarity measure seems to be stable when selecting a reduced number of patches, proving that a few significant patches are enough to represent the image features. This is a consequence of the sparsity of the multiscale transform.

We applied this framework to image retrieval. The proposed approach takes advantage of the properties of its global multiscale descriptors. In particular, it is robust to JPEG2000 compression (i.e. it matches the visual similarity between images with different amounts of blur or compression noise). Retrieval experiments were conducted on two publicly avail-

able datasets of real world images (Nister Recognition Benchmark and the COREL database) to evaluate the average performances of the method. In particular, the Nister dataset was used to benchmark the robustness to several geometric image deformations, such as change of viewpoint, rotation and zoom. Our results showed the reliability of the *SMP* approach with respect to these deformations. Furthermore, although our method is new, its performances tested on two databases are very close to those of several established retrieval methods: a reference retrieval method based on (local) *SIFT* descriptors and two versions of a fuzzy, segmentation-based *UFM* approach: *UFM* and *CLUE* . This indicates that the *SMP* approach adapts to quite different retrieval tasks, from the object level (on the Nister database) to the level of general categories (on the COREL database).

- CHAPTER 3 -

DATA STRUCTURES FOR FAST BREGMAN NEAREST-NEIGHBOR QUERIES

k-nearest neighbor (*k*-NN) search is a crucial tool in many challenging applications of computer vision, ranging from image and information retrieval, to classification and data mining. An example has been presented in Chap. 2, where we have proposed to use a *k*-NN *kernel density estimator* as a basic tool for computing an information-theoretic divergence between sets of image descriptors. Other applications are detailed in the following chapters, where we focus on *k*-NN as a voting rule in the context of supervised classification. Although many efforts have been made to speed up *k*-NN retrieval, this computational task still remains unsatisfactorily expensive and critical in applications. In this chapter, we describe the extension of two well-known data structures for fast indexing and *k*-nearest neighbor retrieval to the information-theoretic class of *Bregman divergences* (symmetrized or not): *metric ball tree* and *vantage point tree*.

On the one hand, we propose an effective Bregman ball tree (*BB-tree*) construction algorithm that adapts locally its "arity" degree to the inherent geometric characteristics of the datasets. As attested by extensive experiments, this allows one to meet more often pruning conditions whenever traversing the tree for *exact* or *approximate* NN queries without yet penalizing the construction time. Since symmetric measures are usually preferred for applications in content-based information retrieval and categorization, we furthermore extend the BB-tree to the case of *symmetrized* Bregman divergences, such as the Jensen-Shannon divergence (*i.e.*, the symmetrical Kullback-Leibler divergence). Exact and approximate 1-NN search experiments using high-dimensional data arising from real-world images (SIFT signatures, color histograms) illustrate that our method improves over the state of the art significantly (up to an order of magnitude).

On the other hand, we generalize the vantage point tree (*vp-tree*) to the same class of distorsion measures. This data structure was originally intro-

duced for information retrieval in metric spaces, and has recently shown very good performances for the retrieval of image patches with respect to the ℓ_2 metric. Later in this chapter, we present our extended algorithm for building and searching for Bregman vantage point trees, and evaluate its performances similarly as for BB-trees.

§ 3.1 Introduction and prior work

Finding *nearest neighbors* (NNs) is a common task occurring in many applications that range from computer vision to machine learning and data mining. Let $\mathcal{S} = \{\boldsymbol{p}_1, ..., \boldsymbol{p}_n\}$ be a set of n d-dimensional data (with d typically ranging up to a few thousand dimensions). Given a *query* point $\boldsymbol{q}$, the nearest neighbor $\mathrm{NN}(\boldsymbol{q})$ is defined as the "closest" point in $\mathcal{S}$ with respect to a dissimilarity measure D:

$$\mathrm{NN}(\boldsymbol{q}) = \arg\min_i D(\boldsymbol{q}, \boldsymbol{p}_i) \ . \tag{3.1}$$

Instead of merely considering the closest neighbor, queries can be enlarged to report the first k "closest" points (*e.g.*, k-NN classification). Furthermore, many applications enable to relax the exact search for getting just *good* approximate nearest neighbors [Cha08b].

Besides the theoretical puzzling questions related to the dreaded curse of dimensionality [Cha08b], practitioners make every endeavor to speed up applications by designing tailored schemes for improving over the naïve linear-time $\mathcal{O}(dn)$ brute-force method (perhaps accelerated by parallelization on GPUs [GDB08]). Among the flourishing literature on NN search techniques[1], we may distinguish two main classes of methods:

- those relying on tree-like space decompositions with *branch-and-bound* queries, such as *kD*-trees, *vantage point* trees [KZN08] and *metric ball* trees [Cay08], and
- those based on mapping techniques [APPK08] (*e.g.*, *locality-sensitive hashing*, random projections).

The former tree-based methods improve over the brute force algorithm by *pruning* sub-trees whenever traversing the trees with queries (*exact* NN) or stopping after visiting a given budget of leaves (*approximate* NN). The latter methods concentrate on reducing dimensions while preserving as much as possible distances and controlling geometrically collisions of hash functions, thus only enabling approximate NN search.

Since the Euclidean distance $D(\boldsymbol{p}, \boldsymbol{q}) = ||\boldsymbol{p} - \boldsymbol{q}|| = \sqrt{\sum_{j=1}^{d} (p^{(j)} - q^{(j)})^2}$ is often inappropriate for meaningfully measuring the proximity of *feature*

[1] See http://simsearch.yury.name/tutorial.html

points arising from image applications, a wide range of distortion measures has been proposed and NN data structures have been first extended to arbitrary metrics (*e.g.*, vantage point trees [KZN08]). However, in order to prune, these NN search methods rely fundamentally on the *triangle inequality* property, which is not satisfied by information-theoretic statistical distances [SVR05], such as the Kullback-Leibler divergence (KL for short)[2]:

$$\mathrm{KL}(\boldsymbol{p}||\boldsymbol{q}) = \sum_{j=1}^{d} p^{(j)} \log \frac{p^{(j)}}{q^{(j)}} \ . \tag{3.2}$$

Cayton has recently extended *metric ball trees* to the broad class of Bregman divergences [Cay08], thus introducing the *Bregman ball tree* (BB-tree), which has been later adapted to range search [Cay09]. A Bregman divergence D_F on vector set $\mathcal{X} \subset \mathbb{R}^d$ is defined for a strictly *convex* and *differentiable* generator $F(\boldsymbol{x}) : \mathcal{X} \subset \mathbb{R}^d \mapsto \mathbb{R}$ as:

$$D_F(\boldsymbol{p}||\boldsymbol{q}) = F(\boldsymbol{p}) - F(\boldsymbol{q}) - (\boldsymbol{p}-\boldsymbol{q})^T \nabla F(\boldsymbol{q}) \ , \tag{3.3}$$

where ∇F denotes the gradient of F. (See Fig. 3.1 for a geometric interpretation.) These divergences (parameterized by a generator F) include all *quadratic distances* (known as Mahalanobis squared distances, which are the only symmetric Bregman divergences and are obtained for $F(\boldsymbol{x}) = \boldsymbol{\Sigma}^{-1}\boldsymbol{x}$, where $\boldsymbol{\Sigma} \succ 0$ is the positive-definite variance-covariance matrix) and the asymmetric *KL divergence* ($F(\boldsymbol{x}) = \sum_{j=1}^{d} x_j \log x_j$, *i.e.*, the negative Shannon entropy), for which $D_F(\boldsymbol{p}||\boldsymbol{q}) \neq D_F(\boldsymbol{q}||\boldsymbol{p})$. Bregman divergences are essential distortion measures as they are provably the *canonical distances* that generalize the Euclidean flat geometry [AN00, NBN07]. Many fundamental algorithms, such as k-means [BMDG05] and PCA, as well as data structures, such as Voronoi diagrams [NBN07], have been generalized to this class of divergences, thus offering meta-algorithms that can work for *any* Bregman divergence. For example, Banerjee et al [BMDG05] have shown that the celebrated Lloyd k-means algorithm extends to (and only to) the class of Bregman divergences, thus unifying various former algorithms. The Bregman k-means hard clustering algorithm proceeds iteratively in two stages by first assigning points to the closest cluster center (centers are initialized by selecting randomly points from the dataset), then updating these k cluster centers by taking their centroids until convergence is reached (Fig. 3.2). Bregman NN queries can be used to improve computationally the assignment stage [NBN07]. Since Bregman divergences are typically non-metric asymmetric distortion measures, one can consider both left/right-sided

[2] Also called relative entropy, discrimination divergence, or information divergence (*I*-divergence for short).

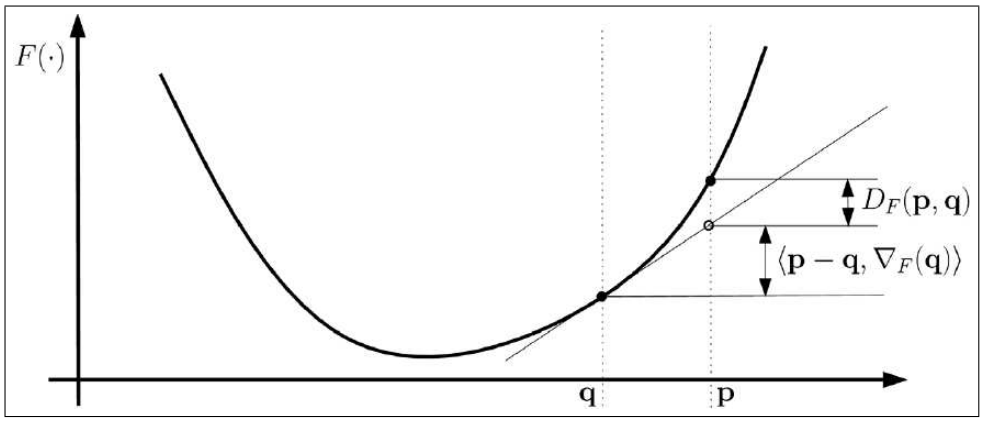


Figure 3.1: *Geometric interpretation of the Bregman divergence as the remainder computed at $\boldsymbol{p}$ of the first order Taylor expansion of F around $\boldsymbol{q}$ [NBN07].*

and symmetrized NN searches, defined as follows:

$$\begin{array}{rcll} \mathrm{NN}_F^r(\boldsymbol{q}) & = & \arg\min_i D_F(\boldsymbol{p}_i||\boldsymbol{q}) & (\mathrm{right}-\mathrm{sided}) \\ \mathrm{NN}_F^l(\boldsymbol{q}) & = & \arg\min_i D_F(\boldsymbol{q}||\boldsymbol{p}_i) & (\mathrm{left}-\mathrm{sided}) \\ \mathrm{NN}_F(\boldsymbol{q}) & = & \arg\min_i (D_F(\boldsymbol{p}_i||\boldsymbol{q}) + D_F(\boldsymbol{q}||\boldsymbol{p}_i))/2 & (\mathrm{symmetrized}) \end{array} \tag{3.4}$$

Besides BB-tree, an alternative data structure for fast NN search is the vantage point tree (*vp-tree*), which was introduced by Yianilos [Yia93] for partitioning a general metric space in a hierarchical way. A tree structure is built by recursively splitting each node covering a set into two *siblings* covering corresponding subsets. Such a node partitioning is based on a randomly chosen *vantage point*. For each point of the node, its distance to the vantage point is compared to a distance threshold. Points with distances smaller than the threshold are classified as "near" and assigned to, say, the left subtree, the remaining others are classified as "far" and assigned to the right subtree. The threshold is usually computed as the *median* of all distances to the vantage point, thus balancing the two sub-trees.

Vp-trees have been rarely but successfully used in applications, such as image indexing [SSF+03] and music information retrieval [SHP08]. In addition, they have recently shown very good performances for image patch retrieval with respect to the ℓ_2 metric [KZN08].

On the one hand, our work takes on the results[3] of Cayton [Cay08], and improves *and* generalizes his Bregman ball-tree (BB-tree) approach. On the other hand, we investigate the use of vantage point tree tailored to the class of Bregman divergences (Bvp-tree). In the following sections, we first summarize the randomized sampling method, which is the reference method for approximate brute-force search and is used for comparisons (Sec. 3.2.1).

[3] C source code and datasets available at http://www.cse.ucsd.edu/~lcayton/

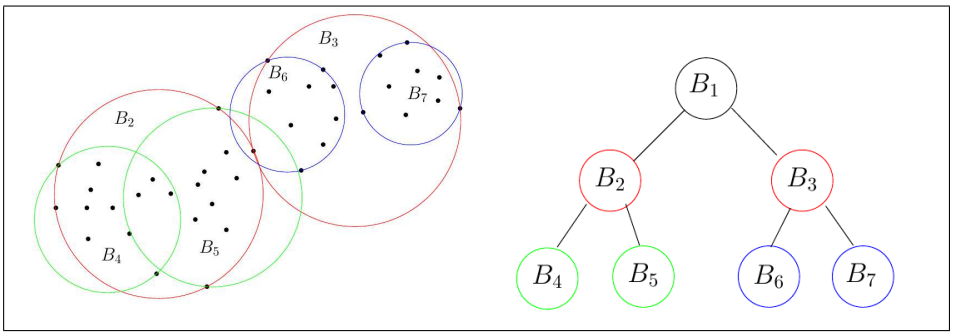


Figure 3.2: *Schema of BB-tree data partitioning by means of Bregman 2-means clustering.*

Then we present our contributions to the BB-tree data structure (Sec. 3.2.2), that are:

- significantly speed up the construction time of Cayton's BB-tree using k-means++ [AV07] and let the tree node arity degree adapts locally to data correlations in order to better balance the tree;
- optimize the projection of queries onto Bregman balls using a novel geodesic bisection search controlled by the Hessian $\nabla^2 F$;
- handle the case of symmetrized Bregman divergences using either explicit or implicit representations for the underlying symmetrized centroids.

Then, we describe our construction and search algorithms for the Bregman vantage point tree (Sec. 3.2.3). Finally, in Sec. 3.3, results of experiments carried out on datasets of high-dimensional SIFT ($d = 1111$) and Corel ($d = 64$) histograms are shown to compare favourably with Cayton's [Cay08] (gain up to a magnitude order).

§ 3.2 k-NN SEARCH VIA BREGMAN DATA STRUCTURES

3.2.1 Randomized sampling for approximate NNs

A simple strategy to significantly speed-up over the brute force NN algorithm consists in sampling the source dataset by choosing to keep a point with probability, say α. At query time, one performs the exact NN query on the sample of mean size αn instead of the full point set, thus obtaining a speed-up factor $\frac{1}{\alpha}$. The impact of sampling the dataset on the NN precision can be measured as the rank $R_{\hat{NN}}$ of the approximated NN, *i.e.*, the true rank of the reported NN in the full set. A probabilistic analysis shows that

the expected value:

$$E\left[R_{\hat{NN}}\right] = \frac{1}{\alpha} - \frac{(1-\alpha)^n[1-\alpha(n+2)]}{\alpha} \tag{3.5}$$

is well-approximated by $\frac{1}{\alpha}$ for large datasets ($n \to \infty$). For a single query the rank is related to the Number Closer (NC) used by Cayton [Cay08], which is defined as the number of points closer to the query than the reported NN. Indeed: $R_{\hat{NN}} = 1 + NC$. For experiments on multiple queries the measure is averaged over all queries, thus providing the Average NN Rank. Note that this approximate NN sampling method applies to any kind of distance measures, and will therefore serve us as the reference basis to compare the efficiency of our improved BB-tree space partitioning method.

3.2.2 Bregman ball trees (BB-trees)

Without loss of generality, we consider only *right-sided* NN queries, as left-sided NN queries can be handled similarly by considering the dual divergence:

$$D_{F^*}(\nabla F(\boldsymbol{q})||\nabla F(\boldsymbol{p})) = D_F(\boldsymbol{p}||\boldsymbol{q}) \ , \tag{3.6}$$

arising from the Legendre conjugate F^* of F, ∇F^* being the functional inverse of ∇F^{-1}. (See [NBN07, Cay08] for details.) Indeed Bregman generators come by pairs (F, F^*), *e.g.*, the dual Legendre conjugates $F(x) = x \log x$ and $F^*(y) = \exp y$ (related by $F'(x) = (F^{*\prime}(y))^{-1}$), see [NBN07].

Outline of Bregman ball trees (BB-trees) Similar to metric ball trees, a BB-tree is built in a top-down fashion by applying recursively a partitioning scheme. First, the root is created to handle the source dataset $\mathcal{S}$. A 2-means (Bregman k-means according to Banerjee et al [BMDG05], with $k = 2$) is computed, splitting $\mathcal{S}$ according to the two centroid points, say $\boldsymbol{c}_l$ and $\boldsymbol{c}_r$, with respect to D_F. These two centroids define a partition $\mathcal{S} = \mathcal{S}_l \cup \mathcal{S}_r$ that can be covered geometrically with corresponding Bregman balls $B(\boldsymbol{c}_l, R_l)$ and $B(\boldsymbol{c}_r, R_r)$, possibly overlapping. (See Fig. 3.2.) This hierarchical decomposition of $\mathcal{S}$ is carried out recursively on $\mathcal{S}_l$ and $\mathcal{S}_r$ until a stopping criterion is eventually met. Such criterion is either a predefined maximum number of points l_0 (stored at leaves) or a prescribed maximum radius r_0. Note that the source points are stored at leaves of BB-tree only, and the subsets $\mathcal{S}_l$ and $\mathcal{S}_r$ may be theoretically unbalanced. Internal nodes store only two left/right Bregman balls covering the point sets stored at their left/right sub-trees.

In both *exact* and *approximate* NN queries, we perform a branch-and-bound search to answer queries. For a given query $\boldsymbol{q}$, the tree is traversed in depth-first-search order from the root to the leaves. At an internal node, we choose to branch first on the sub-tree whose corresponding ball is closer

to the query $\boldsymbol{q}$ (the sibling is temporarily ignored). Once a leaf node is reached, the closest point to $\boldsymbol{q}$ among the points stored at the leaf is computed using the brute-force method. This first visited leaf yields the very first NN point candidate $\boldsymbol{p}'$, thus giving an upper bound $D_F(\boldsymbol{p}'||\boldsymbol{q})$ to the NN distance. In exact search, the tree traversal goes on through all formerly ignored subtrees. In order to decide whether a subtree must be explored or not, we check whether $D_F(\boldsymbol{p}'||\boldsymbol{q}) > \min_{\boldsymbol{x}\in B(\boldsymbol{c},R)} D_F(\boldsymbol{x}||\boldsymbol{q})$, where $\boldsymbol{p}'$ is the current NN candidate. This test is performed by projecting $\boldsymbol{q}$ onto the Bregman ball. The projection of a point $\boldsymbol{q}$ onto a Bregman ball B is the unique minimizer $\boldsymbol{q}_B = \arg\min_{\boldsymbol{x}\in B} D_F(\boldsymbol{x}||\boldsymbol{q})$ [NBN07] (we present an improved algorithm later in this section). Instead of projecting exactly the query point onto the ball, we rather make use of a *Bregman annulus* $B(\boldsymbol{c}, R, R') = \{\boldsymbol{x} \mid R \leq D_F(\boldsymbol{x}||c) \leq R'\}$ that contains the projection by construction: $\boldsymbol{q}_B \in B(\boldsymbol{c}, R, R')$. If $D_F(\boldsymbol{p}'||\boldsymbol{q}) < R$ then the node can be pruned; If $D_F(\boldsymbol{p}'||\boldsymbol{q}) > R'$ then the node must be explored (Fig. 3.3). These lower/upper bounds are computed during the geodesic bisection search of the projection [Cay08]. To conclude, observe that the less the number of pairwise intersecting balls stored at nodes, the better the BB-tree performance. Although exact NN retrieval on BB-trees can often be achieved with much smaller computational cost than the brute-force search, the practical interest of BB-trees is to get significant speed-up search when performing *approximate* NN queries. The approximate search allows one for large speed-up as it stops the branch-and-bound algorithm after exploring a prescribed number of leaves [Cay08].

In the following, we generalize the BB-tree data structure to arbitrary degree ball trees by allowing the algorithm to tune the node arity according to the corresponding subset. Experiments show that this method significantly increases the number of pruned tree nodes.

Speeding up construction time: `BB-tree++` In order to preprocess datasets efficiently by means of Bregman ball trees, instead of running the full regular Bregman k-means algorithm [BMDG05], we just perform a careful light *initialization* of the two cluster centers (*seeds*). Initialization turns out to be the crucial phase of k-means. Indeed, k-means locally optimizes the potential $P(\mathcal{C}) = \sum_{\boldsymbol{p}\in\mathcal{S}} \min_{\boldsymbol{c}\in\mathcal{C}} D_F(\boldsymbol{p}||\boldsymbol{c})$, where $\mathcal{C}$ denotes the set of k centers. Each round assignment/cluster adjustment decreases this potential function so that monotonous convergence is guaranteed. It is striking to know that the worst-case running-time of k-means is theoretically exponential with the dimension although a recent smooth analysis yields polynomial amortized time [MR09]. Moreover, a bad initialization traps k-means into a local optimum. Thus initialization is all the more important, as k-means is called at each internal node of the BB-tree to partition the datasets into two sub-sets. Therefore the idea is to replace the k-means local iterative al-

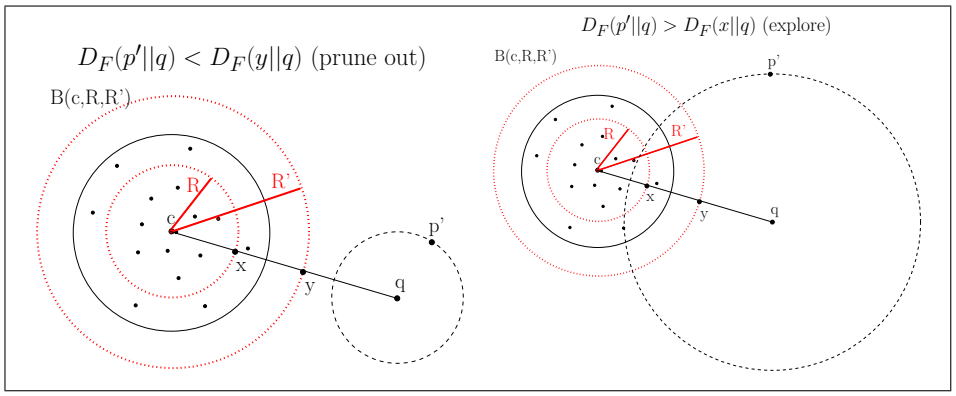


Figure 3.3: *Illustration of the Bregman annulus that allows for branch-and-bound search*

gorithm by a well-chosen initialization that provides a guaranteed upper-bound on the optimal partition. This quantum leap for k-means was discovered by Arthur and Vassilvitskii [AV07] and later extended to Bregman divergences by Nock et al [NLK08].

In order to initialize the seeds, we first draw $\boldsymbol{c}_l$ (cluster "center") uniformly at random among the points of the dataset. Then, we compute the Bregman divergence of this point to all points of $\mathcal{S}$, and draw the second seed $\boldsymbol{c}_r$ according to the divergence distribution:

$$\pi_i = \frac{D_F(\boldsymbol{p}_i||\boldsymbol{c}_l)}{\sum_{\boldsymbol{p}_j \in \mathcal{S}} D_F(\boldsymbol{p}_j||\boldsymbol{c}_l)} \ . \tag{3.7}$$

Thus $\boldsymbol{c}_r$ can never be $\boldsymbol{c}_l$, since $D_F(\boldsymbol{c}_l||\boldsymbol{c}_l) = 0$; therefore the probability of drawing $\boldsymbol{c}_l$ again is zero. This careful initialization yields an extremely fast tree construction, which statistically provides nice splitting and tends to balance subsets stored at leaves. Arthur and Vassilvitskii [AV07] proved that this initialization yields a k-means score $P(\mathcal{C})$ at most $8(2 + \log k)$ of the optimal value. Similar bounds in $\mathcal{O}(\log k)$ for generic Bregman divergences were later reported by Nock *et al* [NLK08]. Note that the two left/right Bregman balls stored at internal nodes tend to minimize the Bregman information [BMDG05] (*i.e.*, variance for the square potential $F(x) = x^2$, mutual information for the negative Shannon entropy $F(x) = x \log x$, etc). Next, we improve the partition at each internal node by *learning* its degree (number of siblings) from the local datasets.

Learning the tree branching factor Answering queries may range from optimal logarithmic time (*i.e.*, the shortest path to a leaf) up to linear time for a complete tree traversal. Therefore, it is important to partition the source datasets into as many as possible non-overlapping Bregman balls.

However, consider a dataset consisting of three separated Gaussian point samples. Forcing this set to be split into two will likely create overlapping balls. Thus we better learn the number of sub-sets when partitioning the data so that the induced Bregman balls better fit the intrinsic geometric characteristics of the set (refer to Fig. 3.4). We propose to use the G-means strategy [HE03] for adapting the branching factor bf_i (up to a maximum branching factor BF) of each internal node of the BB-tree to the underlying distribution of the data. The self-tuned splitting algorithm for internal nodes is summarized in Alg. 1. The underlying idea is to assume Gaussian distribution of each group of points (hence the name G-means). (This is not so restrictive, as any smooth density function may be reasonably well approximated using a Gaussian mixture model.) Our use of G-means [HE03] algorithm starts by setting $k = 2$ and then test for Gaussian distribution of the points using the Anderson-Darling statistical test. (This merely requires sorting the data projected on a line, *i.e.*, scalar values.) Given a confidence level α, if the Anderson-Darling test returns `true`, we keep the center, otherwise we split it into two. Between two rounds, we simply run Bregman k-means++ initialization on the dataset and get all the new centers that hopefully refine the partitioning. We enforce a maximum degree to each internal node in order to strike a balance between the average tree depth and the overall ball tree shape.

The initialization to k clusters follows the same principle. Namely, we draw the l-th seed from the dataset uniformly according to the distribution:

$$\pi_i = \frac{D_F(\boldsymbol{p}_i||\mathcal{C}_{l-1})}{\sum_{\boldsymbol{p}_j \in \mathcal{S}} D_F(\boldsymbol{p}_j||\mathcal{C}_{l-1})} \ , \tag{3.8}$$

where $\mathcal{C}_{l-1}$ denotes the formerly chosen $(l-1)$-th seed and $D_F(\boldsymbol{p}||\mathcal{S}) = \min_{\boldsymbol{x} \in \mathcal{S}} D_F(\boldsymbol{p}||\boldsymbol{x})$.

When visiting the BB-tree for answering nearest neighbor queries, we use a *priority queue*. In order to relax the exact NN to approximate NN queries, the criterion of stopping the search once a few leaves have been visited was proposed and proven successful by Cayton [Cay08]. However, there was no guarantee to get a good approximation to the exact NN because leaves containing subsets that are close to the exact NN are not necessarily close in the tree. We improve this point by a careful non-recursive implementation of the tree traversal, which allows us to order the nodes to be explored by their divergence to the query point (*i.e.*, the divergence of the query to the ball centers). Using a priority queue guarantees that nearest nodes are always explored first when traversing back the tree, by jumping appropriately to the most likely not yet visited sub-tree.

Algorithm 1 Bregman Ball Partition $(B(\boldsymbol{c}, R),\ BF,\ R_{min})$

Input: Bregman Ball $B(\boldsymbol{c}, R)$, integer $BF > 0$ (max branching factor), real R_{min} (min ball radius);
SET $childrenList$ = EMPTY, $tempQueue$ = EMPTY;
SET $bf = 0$ (branching factor);
$B_l(\boldsymbol{c}_l, R_l), B_r(\boldsymbol{c}_r, R_r) \leftarrow$ Bregman 2-Means$(B(\boldsymbol{c}, R))$;
if $R_l > R_r$ **then**
  PUSH $(B_l,\ tempQueue)$; PUSH $(B_r,\ tempQueue)$;
else
  PUSH $(B_r,\ tempQueue)$; PUSH $(B_l,\ tempQueue)$;
end if
INCREMENT bf BY 2;
while $tempQueue$ NOT EMPTY **do**
  $B_t(\boldsymbol{c}_t, R_t) \leftarrow$ POP$(tempQueue)$;
  if $bf + 1 \leq BF$ AND $R_t > R_{min}$ **then**
    $B_l,\ B_r \leftarrow$ Bregman 2-Means$(B_t(\boldsymbol{c}_t, R_t))$; $NeedToSplit \leftarrow$ Anderson-Darling Test (B_l, B_r);
    if $NeedToSplit$ = FALSE **then**
      PUSH $(B_t(\boldsymbol{c}_t, R_t),\ childrenList)$;
    else
      PUSH $(B_l,\ tempQueue)$;
      PUSH $(B_r,\ tempQueue)$;
      INCREMENT bf BY 1;
    end if
  else
    PUSH $B_t(\boldsymbol{c}_t, R_t),\ childrenList)$;
  end if
end while
Output: $childrenList, bf$

Revisiting the geodesic bisection search We propose an improved *geodesic bisection* search method controlled by the variation of the Bregman generator, namely its Hessian $\nabla^2 F$. Let us recall that the geodesic Γ_{pq} linking $\boldsymbol{p}$ to $\boldsymbol{q}$ is defined as $\Gamma_{pq} = \{\mathrm{LERP}(\lambda, \boldsymbol{p}, \boldsymbol{q}) \mid \lambda \in \mathbb{R}\}$, where:

$$\mathrm{LERP}(\lambda, \boldsymbol{p}, \boldsymbol{q}) = \nabla F^{-1}((1-\lambda)\nabla F(\boldsymbol{p}) + \lambda \nabla F(\boldsymbol{q}))\ . \tag{3.9}$$

(See [NBN07, Cay08, NN09a] for the details.)

In order to find the Bregman projection $\boldsymbol{q}_B = \arg\min_{\boldsymbol{x} \in B(\boldsymbol{c}, R)} D_F(\boldsymbol{x}||\boldsymbol{c})$ of a query point $\boldsymbol{q}$ onto a Bregman ball $B(\boldsymbol{c}, R)$, we first check whether $\boldsymbol{q}$ is outside the ball or not: $D_F(\boldsymbol{q}||\boldsymbol{c}) > R$. If not, $D_F(\boldsymbol{q}||\boldsymbol{c}) \leq R$ and the projection is simply the point itself: $\boldsymbol{q}_B = \boldsymbol{q}$. Let us consider the geodesic segment $[\boldsymbol{cq}]$, the projection point $\boldsymbol{q}_B$ belongs necessarily to the geodesic

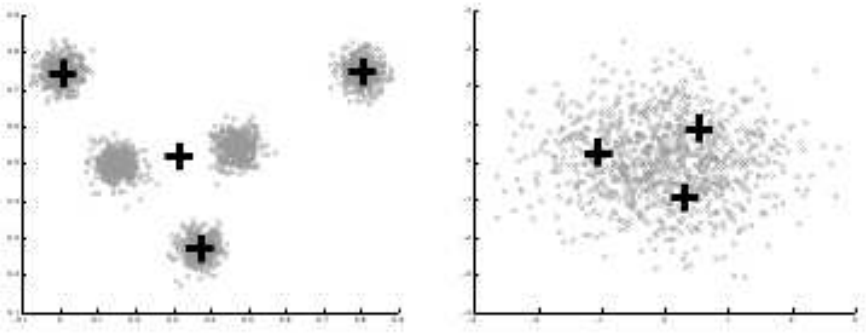


Figure 3.4: *Two examples of improperly choosing the value of k for k-means clustering [HE03].*

Γ_{cq}: $\boldsymbol{q}_B = \mathrm{LERP}(\lambda_q, \boldsymbol{c}, \boldsymbol{q})$ for some $\lambda_q \in [0,1]$. The value λ_q is approximated by a bisection search as follows. Initially, let $\lambda_m^{(0)} = 0$ and $\lambda_M^{(0)} = 1$ and consider a midpoint $\boldsymbol{m} = \mathrm{LERP}(\lambda^{(0)}, \boldsymbol{c}, \boldsymbol{q})$ obtained for $\lambda^{(0)} = \frac{\lambda_m^{(0)} + \lambda_M^{(0)}}{2}$. If $D_F(\boldsymbol{m}||\boldsymbol{c}) > R$ then recurse on $[\lambda_m^{(1)}, \lambda_M^{(1)}] = [\lambda_m^{(0)}, \lambda^{(0)}]$, otherwise recurse on $[\lambda_m^{(1)}, \lambda_M^{(1)}] = [\lambda^{(0)}, \lambda_M^{(0)}]$ and so on until the range size $\lambda_M^{(i)} - \lambda_m^{(i)}$ goes below a threshold ϵ (typically $\epsilon \in [10^{-10}, 10^{-5}]$). This yields a fine approximation of $\boldsymbol{q}_B \sim \mathrm{LERP}(\lambda^{(i)}, \boldsymbol{c}, \boldsymbol{q})$ by splitting the "λ" intervals a dozen of times. (Recall that the algorithm keeps a Bregman annulus, and refine the inner/outer radii to decide whether to prune or explore a sub-tree.) Halving the geodesic by halving the λ coefficients is however *sub-optimal* since the two portions $[\boldsymbol{qm}]$ and $[\boldsymbol{mc}]$ of the geodesic splitted by $\boldsymbol{m}$ may have significantly different divergences (*e.g.*, $D_F(\boldsymbol{q}||\boldsymbol{m}) \neq D_F(\boldsymbol{m}||\boldsymbol{c})$ for KL).

There are several ways of formalizing the geodesic cut yielding to different formulas; here we present the method that worked best experimentally. We split the geodesic $[\boldsymbol{pq}]$ in half amounts to seek for λ such that:

$$D_F(\boldsymbol{p}||\mathrm{LERP}(\lambda, \boldsymbol{p}, \boldsymbol{q})) = D_F(\mathrm{LERP}(\lambda, \boldsymbol{p}, \boldsymbol{q})||\boldsymbol{q}) \ . \tag{3.10}$$

The Bregman divergence can be interpreted as the exact remainder of the Taylor expansion of generator F [AN00]:

$$D_F(\boldsymbol{p}||\boldsymbol{q}) = \frac{1}{2}(\boldsymbol{p}-\boldsymbol{q})^T \nabla^2 F(\varepsilon)(\boldsymbol{p}-\boldsymbol{q}) \tag{3.11}$$

for some $\varepsilon \in [\boldsymbol{pq}]$. Thus it follows that:

$$D_F(\boldsymbol{p}||\boldsymbol{q}) \simeq \frac{1}{2}(\boldsymbol{p}-\boldsymbol{q})^T \nabla^2 F(\varepsilon)(\boldsymbol{p}-\boldsymbol{q}) \tag{3.12}$$

for *any* $\varepsilon \in [\boldsymbol{pq}]$. For close points $\boldsymbol{p}$ and $\boldsymbol{q}$, we can thus approximate as:

$$D_F(\boldsymbol{p}||\boldsymbol{q}) \simeq \frac{1}{2}(\boldsymbol{p}-\boldsymbol{q})^T \nabla^2 F(\boldsymbol{p})(\boldsymbol{p}-\boldsymbol{q}) \ , \tag{3.13}$$

or:

$$D_F(\boldsymbol{p}||\boldsymbol{q}) \simeq \frac{1}{2}(\boldsymbol{p}-\boldsymbol{q})^T \nabla^2 F(\boldsymbol{q})(\boldsymbol{p}-\boldsymbol{q}) \ . \tag{3.14}$$

For the sake of simplicity, let us consider 1D Bregman divergences so that $\nabla^2 F = f''$. It follows from $D_F(\boldsymbol{p}||\mathrm{LERP}(\lambda, \boldsymbol{p}, \boldsymbol{q})) = D_F(\mathrm{LERP}(\lambda, \boldsymbol{p}, \boldsymbol{q})||\boldsymbol{q})$ that:

$$\frac{1}{2}\lambda^2(\boldsymbol{p}-\boldsymbol{q})f''(\boldsymbol{p}) \simeq \frac{1}{2}(1-\lambda)^2(\boldsymbol{p}-\boldsymbol{q})f''(\boldsymbol{q}) \ . \tag{3.15}$$

Hence:

$$(\frac{\lambda}{1-\lambda})^2 \simeq \frac{f''(\boldsymbol{q})}{f''(\boldsymbol{p})} \tag{3.16}$$

and, solving for λ, we get:

$$\lambda = \frac{c}{c+1} \ , \tag{3.17}$$

with $c \simeq \sqrt{\frac{f''(\boldsymbol{q})}{f''(\boldsymbol{p})}}$. (For multivariate Bregman divergences, we let $c \simeq \sqrt{\frac{||\nabla^2 F(\boldsymbol{q})||}{||\nabla^2 F(\boldsymbol{p})||}}$.) Note that λ is guaranteed to fall always in $[0,1]$. For squared Euclidean distance, the Hessian is constant $f''(x) = 2$ and for the KL divergence, it is quickly varying as $x \to 0^+$: $f''(x) = \frac{1}{x}$. Thus for two points $\boldsymbol{p}$ and $\boldsymbol{q}$ in the KL geometry with $\boldsymbol{p} \to 0$, λ tends to 0 and not $\frac{1}{2}$, as wished. For squared Euclidean distance, this yields the usual dichotomic search ($c = 1$): $\lambda = \frac{1}{2}$. The Hessian-based cut method reduces the number of iteration steps for estimating the projection point $\boldsymbol{q}_B$. Note that this technique also applies for walking efficiently toward any given point on geodesics, and can thus be used to improve the computation of symmetrized Bregman centroid, as described in [NN09a]. Furthermore, this method can be used for checking whether two Bregman balls intersect or not (a primitive that is also required for handling Bregman vantage-point trees).

Handling symmetrized Bregman divergences Content-based information retrieval and categorization systems often need the distortions measures to be symmetric [SVR05]. Except for generalized quadratic (Mahalanobis) distances, the symmetrized Bregman divergence is technically not a Bregman divergence [NBN07]. An example is given by the symmetric KL divergence (SKL), which goes by the name of Jensen-Shannon:

$$\mathrm{JS}(\boldsymbol{p};\boldsymbol{q}) = \frac{1}{2}\mathrm{KL}\left(\boldsymbol{p}||\frac{\boldsymbol{p}+\boldsymbol{q}}{2}\right) + \frac{1}{2}\mathrm{KL}\left(\boldsymbol{q}||\frac{\boldsymbol{p}+\boldsymbol{q}}{2}\right) \ . \tag{3.18}$$

It turns out that such symmetrized Bregman divergences [NN09a] are all generalizations of the Jensen remainder obtained for convex generators F[4]:

$$\mathrm{JS}_F(\boldsymbol{p};\boldsymbol{q}) = \frac{1}{2}(F(\boldsymbol{p}) + F(\boldsymbol{q})) - F\left(\frac{\boldsymbol{p}+\boldsymbol{q}}{2}\right) \ . \tag{3.19}$$

The symmetrized Bregman divergences can also be interpreted as a special case of *mixed* Bregman divergences, for which a remarkable extension of Bregman k-means++ has been reported [NLK08].

The mixed Bregman divergence is defined for a value $\alpha \in [0,1]$ and triplets of points as follows:

$$D_{F,\alpha}(\boldsymbol{l}||\boldsymbol{x}||\boldsymbol{r}) = (1-\alpha)D_F(\boldsymbol{l}||\boldsymbol{x}) + \alpha D_F(\boldsymbol{x}||\boldsymbol{r}) \ . \tag{3.20}$$

It thus includes the regular Bregman divergence ($\alpha = 0$), the inversely oriented divergence ($\alpha = 1$) and the symmetrized Bregman divergence ($\alpha = \frac{1}{2}$ with $\boldsymbol{l} = \boldsymbol{r}$). The construction of *mixed Bregman ball trees* follows similar principles but instead of storing for each ball one center, we store two left and right centers. Alternatively, the symmetrized Bregman centroid may also be explicitly computed following the geodesic-walk algorithm of [NBN07]. Indeed, the geodesic of symmetrized Bregman divergences follows the same expression: $\Gamma_{\boldsymbol{pq}} = \{\mathrm{LERP}(\lambda, \boldsymbol{p}, \boldsymbol{q}) \mid \lambda \in \mathbb{R}\}$.

Sided or symmetrized Bregman divergence? SKL has already proven useful and appropriate for numerous applications including image retrieval applications (*e.g.*, see [SVR05]). For asymmetric Bregman divergences like KL, we still have to choose between either the left-sided or right-sided query type. This choice depends on applications, as they exhibit different mathematical properties, and thus report different exact/approximate NNs in practice. The right-sided query:

$$\arg\min_{\boldsymbol{p}_i \in \mathcal{S}} \mathrm{KL}(\boldsymbol{p}_i||\boldsymbol{q}) = \arg\min_{\boldsymbol{p}_i \in \mathcal{S}} \sum_{j=1}^{d} p_i^{(j)} \log \frac{p_i^{(j)}}{q^{(j)}} \tag{3.21}$$

is *zero-looking*: whenever a component $q^{(j)}$ of the query is close to zero, it seeks for points $\boldsymbol{p}_i$ with $p_i^{(j)}$ close to zero too (otherwise KL grows fast and is potentially unbounded for $q^{(j)} = 0$; but JS is always bounded). Similarly, for left-sided query:

$$\arg\min_{\boldsymbol{p}_i \in \mathcal{S}} \sum_{j=1}^{d} q^{(j)} \log \frac{q^{(j)}}{p^{(j)}} \ , \tag{3.22}$$

[4]Besides the Jensen-Shannon divergence for $F(x) = x \log x$, these so-called Burbea-Rao divergences also include the important COSH distance used in speech/sound processing for the Burg entropy $F(x) = -\log x$. These divergences are always bounded (cannot tend to infinity).

the left-sided NN is *zero-discarding*, as the component $p^{(j)}$ is not taken into account for $q^{(j)} \to 0$.

Banerjee et al [BMDG05] showed a bijection between regular exponential families in statistics and (regular) Bregman divergences. This gives a way to *design* the most appropriate Bregman divergence from signal modeling. For example, the sound spectra tends to follow exponential distributions of the form $p(x) = \lambda e^{-\lambda x}$, and thus the corresponding divergence is obtained for the Burg entropy $F(x) = -\log x$. It is precisely the well-known discrete Itakura-Saito divergence $\mathrm{IS}(p||q) = \frac{p}{q} + \log \frac{q}{p} - 1$ that worked best experimentally for sound/speech processing. Note that the KL divergence on d-dimensional point sets can be computed as the KL divergence of corresponding exponential families [NN09a]. Indeed, d-dimensional probability vectors (*e.g.*, normalized histograms) can be considered as multinomial distributions with $d-1$ degrees of freedom. Therefore the BB-tree++ for KL can be built not on the source data points, but rather on the $(d-1)$-dimensional "natural" points for the generator $F(x) = \log(1+\exp x)$, the logistic entropy (see [NN09a] for related details).

3.2.3 Bregman Vantage-point Trees

In this section we first describe the *Bvp-tree* data structure based on the seminal description of Yianilos' vp-tree [Yia93], then we describe our algorithm for checking the pruning conditions when visiting the tree, which is our main contribution for extending this data structure to Bregman divergences.

Outline of Bregman vantage point tree (`Bvp-tree`) A Bvp-tree is built by applying recursively the same partitioning procedure as described in [Yia93], except for replacing distances by Bregman divergences. First, the tree root is created, which represents the whole dataset $\mathcal{S}$. Then a vantage point v is randomly drawn from the dataset and the distance $D_F(\boldsymbol{x}||\boldsymbol{v})$ is computed for each point $\boldsymbol{x} \in \mathcal{S}$. A distance threshold τ defines a partition $\mathcal{S} = \mathcal{S}_l \cup \mathcal{S}_r$, where S_l is covered by the Bregman ball $B(\boldsymbol{v}, \tau) = \{\boldsymbol{x} \mid D_F(\boldsymbol{x}||\boldsymbol{v}) < \tau\}$ and $\mathcal{S}_r$ is contained in the complement subset $\mathcal{S} \setminus B(\boldsymbol{v}, \tau) = \{\boldsymbol{x} \mid D_F(\boldsymbol{x}||\boldsymbol{v}) \geq \tau\}$. In our implementation, we define τ as the median of the distances $\{D_F(\boldsymbol{x}||\boldsymbol{v}) \mid \boldsymbol{x}\}$. This hierarchical decomposition of $\mathcal{S}$ is applied recursively on $\mathcal{S}_l$ and $\mathcal{S}_r$ until a stopping criterion is eventually met. This criterion is typically based either on setting: (1) the maximum number of leaf points bs (*bucket size*), or (2) the maximum leaf radius r_0. When such a criterion is met, two leaves are created to store the corresponding source points. (All internal nodes store only Bregman balls $B(\boldsymbol{v}, \tau)$.) Figure 3.5 shows an example of the space partition induced by a Bvp-tree.

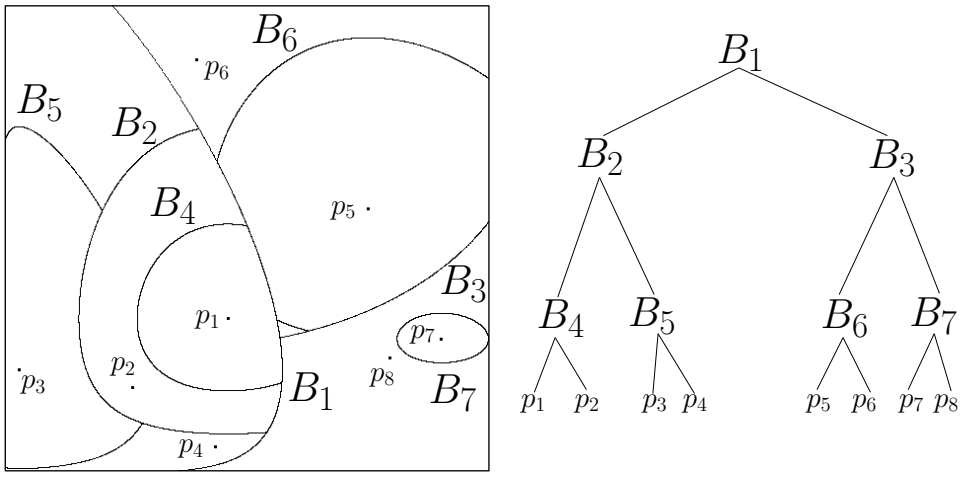


Figure 3.5: *Schematic description of Bvp-tree construction for a set of* 8 *points (wrt. SKL aka Jensen-Shannon divergence).*

In order to retrieve the NN for a given query $\boldsymbol{q}$, we perform a branch-and-bound traversal of the tree. The tree is visited by traversing it from the root to the leaves following a given branching order (depth first search). Namely, at any internal node, we choose to branch first on the sub-tree whose corresponding ball is *closer* to the query $\boldsymbol{q}$. Temporarily ignored siblings are added to a *priority queue* for successive exploration. (The smaller the distance of a sibling node to the query point, the largest its priority.) The first visited leaf yields the very first NN candidate point $\boldsymbol{p}'$, thus giving an upper bound $D_F(\boldsymbol{p}'||\boldsymbol{q})$ to the NN distance. Indeed, the true NN cannot lie outside the boundary of the Bregman ball $B(\boldsymbol{q}, D_F(\boldsymbol{p}'||\boldsymbol{q}))$ (query ball). Then, in exact search, all formerly ignored subtrees are explored according to their priority order. Each subtree needs to be visited or not depending on whether $D_F(\boldsymbol{p}'||\boldsymbol{q}) > \min_{\boldsymbol{x}\in B(\boldsymbol{c},R)} D_F(\boldsymbol{x}||\boldsymbol{q})$, where $\boldsymbol{p}'$ is the current NN candidate. This test is performed by checking whether the ball $B(c, R)$ intersects the current query ball $B(\boldsymbol{p}', D_F(\boldsymbol{p}'||\boldsymbol{q}))$ or not. If the two balls do not intersect, then the node can be pruned. Otherwise, it must be explored. Every time a new NN candidate is found, the upper bound is updated, thus progressively reducing the size of the search subspace. Pruning subspaces is a major advantage of such data structures, since the more leaves are pruned out, the more significant is the computational speed-up over brute-force search.

Pruning condition In order to check whether the intersection between two Bregman balls is void or not, we propose a test that enables us to prune nodes that cannot contain a closer neighbor than the actual NN candidate. Indeed, given a query point $\boldsymbol{q}$ and a NN candidate $\boldsymbol{p}'$, the Bregman ball

$B(\boldsymbol{q}, r')$ of radius $r' = D_F(\boldsymbol{p}'||\boldsymbol{q})$ centered at $\boldsymbol{q}$ defines the only space region where a closer NN may be found. As a result, any subtree rooted at a node that does not intersect $B(\boldsymbol{q}, r')$ can be pruned without impacting the retrieval precision.

Consider two non-concentric Bregman balls $B_1(\boldsymbol{p}, r_p)$ and $B_2(\boldsymbol{q}, r_q)$. The *radical hyperplane* H_{12} is the locus of points that have equal power with respect to these two balls. It is defined as follows:

$$H_{12}: \; B_1(\boldsymbol{x}) - B_2(\boldsymbol{x}) = 0 \;, \tag{3.23}$$

where:

$$B_1(\boldsymbol{x}) \;:\; D_F(\boldsymbol{x}||\boldsymbol{p}) - r_p = 0 \tag{3.24}$$

$$B_2(\boldsymbol{x}) \;:\; D_F(\boldsymbol{x}||\boldsymbol{q}) - r_q = 0 \tag{3.25}$$

are the power equations of the two balls. Plugging the definition of $D_F(\boldsymbol{x}||\cdot)$ yields the following equation of the radical hyperplane H_{12}:

$$F(\boldsymbol{q}) - F(\boldsymbol{p}) + r_2 - r_1 + \langle \boldsymbol{x}, \nabla_F(\boldsymbol{q}) - \nabla_F(\boldsymbol{p}) \rangle + \langle \boldsymbol{p}, \nabla_F(\boldsymbol{p}) \rangle - \langle \boldsymbol{q}, \nabla_F(\boldsymbol{q}) \rangle = 0 \,, \tag{3.26}$$

where $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ denotes the vector inner product $\boldsymbol{x}^\mathsf{T} \boldsymbol{y}$. If the balls intersect, then the radical hyperplane must necessarily contain points of intersection (Fig. 3.6).

In order to test this condition, we propose to check the point of intersection between the radical hyperplane and the geodesic $\Gamma_{\boldsymbol{pq}}$ linking $\boldsymbol{p}$ to $\boldsymbol{q}$, which is defined as: $\Gamma_{\boldsymbol{pq}} = \{\mathrm{LERP}(\lambda, \boldsymbol{p}, \boldsymbol{q}) \mid \lambda \in \mathbb{R}\}$. (See Eq. 3.9.) If neither ball contains this point, then their intersection is empty. We implement this test as a bisection geodesic walk algorithm similar to that used in [NN09a] for computing symmetrized Bregman centroids. Note that we first check whether a ball contains the center of the other, as this would make more iterations useless. Furthermore, we use bounds to stop the bisection algorithm as soon as possible, without need for precisely computing the common intersection point (Alg. 2).

§ 3.3 Experiments

We carried out extensive experiments on the BB-tree++ and Bvp-tree data structures by collecting various statistics for both KL and SKL queries. First we focused on evaluating performances of both construction and search of our adaptive ball trees, comparing them with the regular `BB-tree` of Cayton [Cay08]. We used the same baseline C code and datasets so that the presented results are as fair as possible [5]. When a direct comparison

[5] We thank Lawrence Cayton (UCSD) for sharing with us both his code and SIFT/Corel high-dimensional datasets.

Algorithm 2 Bregman Ball Intersection$(B_q(\boldsymbol{q}, R_q), B_v(\boldsymbol{v}, R_v), \lambda_l, \lambda_r)$

Input: Bregman Balls $B_q(\boldsymbol{q}, R_q)$ (query point ball), $B_v(\boldsymbol{v}, R_v)$ (vantage point ball), $\lambda_l, \lambda_r \in (0, 1)$;
SET $\boldsymbol{x}_{\lambda_l} = \nabla F^{-1}((1-\lambda_l)\nabla F(\boldsymbol{q}) + \lambda_l \nabla F(\boldsymbol{v}))$;
SET $x_{\lambda_r} = \nabla F^{-1}((1-\lambda_r)\nabla F(\boldsymbol{q}) + \lambda_r \nabla F(v))$;
if $D_F(\boldsymbol{x}_{\lambda_r}||\boldsymbol{q}) < R_q$ OR $D_F(\boldsymbol{x}_{\lambda_l}||\boldsymbol{v}) < R_v$ **then**
  return YES;
end if
SET $\lambda = \frac{\lambda_l + \lambda_r}{2}$;
SET $\boldsymbol{x}_\lambda = \nabla F^{-1}((1-\lambda)\nabla F(\boldsymbol{q}) + \lambda \nabla F(\boldsymbol{v}))$;
if $D_F(\boldsymbol{x}_\lambda||\boldsymbol{q}) > R_q$ AND $D_F(\boldsymbol{x}_\lambda||\boldsymbol{v}) > R_v$ **then**
  return NO;
else if $D_F(\boldsymbol{x}_\lambda||\boldsymbol{q}) < R_q$ AND $D_F(\boldsymbol{x}_\lambda||\boldsymbol{v}) < R_v$ **then**
  return YES;
else if $D_F(\boldsymbol{x}_\lambda||\boldsymbol{q}) < R_q$ AND $D_F(\boldsymbol{x}_\lambda||\boldsymbol{v}) > R_v$ **then**
  SET $\lambda_l = \lambda$;
  return Ball Intersection$(B_q(\boldsymbol{q}, R_q), B_v(\boldsymbol{v}, R_v), \lambda_l, \lambda_r)$;
else if $D_F(\boldsymbol{x}_\lambda||\boldsymbol{q}) > R_q$ AND $D_F(\boldsymbol{x}_\lambda||\boldsymbol{v}) < R_v$ **then**
  SET $\lambda_r = \lambda$;
  return Ball Intersection$(B_q(\boldsymbol{q}, R_q), B_v(\boldsymbol{v}, R_v), \lambda_l, \lambda_r)$;
end if

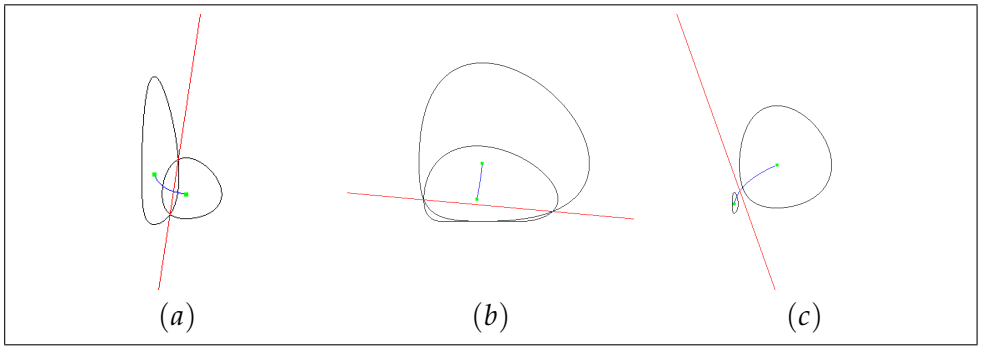


Figure 3.6: *Examples radical hyperplanes for two (a,b) intersecting, and (c) non-intersecting Bregman balls.*

to Cayton's results was not possible (namely for symmetrized NN queries) we compared our results to those of the Randomized Sampling method (Sec. 3.2.1).

Then, we carried out experiments on Bvp-trees and evaluated the performances of this data structure when varying the partitioning termination criterion and the computational cost of construction, as well as the performances of searching for both *exact* and *approximate* NN.

For this purpose, we used two feature sets originating from databases of images, each set containing two parts: (1) a source point database, which is used for building the tree structure at pre-processing time, and (2) a query point set, which is used for performing on-line queries on the tree.

- The first dataset contains high-dimensional *histograms of SIFT* descriptors ($d = 1111$), *i.e.*, Bag-of-Features descriptors [SZ03], extracted from images of the Pascal 2007 challenge [EGW+]. Such descriptors are widely used in applications of image retrieval and categorization, which generally require fast and precise nearest neighbor search [Low04]. The SIFT dataset contains $10,000$ reference points and $2,300$ query points.

- The second dataset is a collection of *color histograms* obtained from Corel images. This dataset has been extensively used for benchmarking several computer vision applications. It contains $60,000$ reference points and $6,616$ query points ($d = 64$).

In the following we present the most significant results of both *exact* and *approximate* NN search. First, we show in Sec. 3.3.1 the typical computational speed-up obtained by `BB-tree++` for the construction stage. We stress out the experimental properties of the ball trees obtained by gathering various statistics when varying the maximum branching factor and the partitioning termination criterion. Then we discuss NN retrieval results in Sec. 3.3.2, specially focusing on approximate search using the asymmetric KL divergence. In particular, we comment the experimental trend of retrieval performances when varying the branch-and-bound stopping criterion. These results are compared to those reported by Cayton [Cay08] for the same datasets. Finally we present some novel results on symmetrized and mixed Bregman queries, which significantly outperform the Random Sampling approach, thus validating the effectiveness of our adaptive BB-trees.

3.3.1 BB-tree construction

First of all we analyze the typical profile of `BB-tree++` as a function of two fundamental construction parameters:

- the maximum number of points contained in a leaf (*bucket size*), and
- the maximum branching factor of an interior node BF. (See Alg. 1.)

These parameters determine the conditions for stopping the recursion when constructing a BB-tree. We used different values of the construction parameters in order to highlight the benefits in terms of computational cost and adaptivity to data distribution. In Table 3.1, we report results of making the bucket size, bs, vary between 50 and 200, while relaxing the branching condition for ball partitioning from $BF = 2$ to $BF = 10$. For each combination of these two parameters, we measured the main properties that describe the tree structure. In particular we give the maximum and average *tree depth*, the average *branching factor* of interior nodes, the *number of leaves* and the *tree size* (*i.e.*, the overall number of tree nodes). As expected, the tree size considerably depends on bs. However, for any fixed value of bs, the tree size shows only slight variations as a function of the branching factor bf, whereas the number of leaves increases significantly with bf itself. Indeed, increasing the branching factor tends to reduce the average tree depth and concentrate more nodes at the lowest levels of the tree, specially at the leaf level. The reason is clear when considering the ratio between the tree size and the number of leaves: the larger the branching factor, the larger the number of leaves with respect to the overall size. Consequently, the computational cost of BB-tree construction is generally smaller, as it is roughly proportional to the number of internal nodes. More precisely, partitioning a node N_i containing n_i points with a branching factor bf_i requires $2n_i(bf_i - 1)$ divergence computations, which can be asymptotically approximated by $\mathcal{O}(2n_i)$, as variations of bf_i are neglectable compared to those of n_i. Since this term represents the main contribution to the overall construction cost, it justifies the benefit of reducing the average depth of the tree, thus reducing the number of nodes to be partitioned. In addition, having fewer interior nodes in a BB-tree generally allows for faster search, as it is shown in Table 3.3 for the same dataset as in Table 3.1. Indeed, pruning out a subtree in this case implies discarding more leaves, hence reducing the overall number of branch-and-bound iterations.

We also evaluated performances of the Bregman 2-`means++` partitioning method described in Sec. 3.2.2. Namely, Tab. 3.2 compares the BB-tree construction results of our method (third row) to those obtained with up to 10 assigment/relocation iterations of 2-means clustering, using both classic seeding (first column) and "careful" seeding (second column). (Results are averaged over a number of trials.) Although the two full clustering algorithms do not differ significantly, neither in terms of tree characteristics nor

BF	*bs*	*depth*	$depth_{avg}$	bf_{avg}	*nLeaves*	*size*
2	**50**	**24**	**14.09**	**2.00**	**2271**	**4541**
2	100	24	13.78	2.00	1189	2377
2	200	20	11.47	2.00	592	1183
3	50	19	9.31	2.49	2605	4348
3	100	15	8.29	2.63	1300	2098
3	200	13	6.99	2.75	655	1028
4	**50**	**16**	**7.90**	**2.80**	**2818**	**4385**
4	100	17	6.96	3.07	1469	2178
4	200	10	6.09	3.48	818	1147
10	50	12	5.81	3.04	2853	4251
10	100	10	4.64	3.86	1637	2209
10	200	9	3.93	5.13	997	1238

Table 3.1: *BB-tree construction results for the Corel dataset . BF and bs denote respectively the maximum branching factor and the bucket size. BB-tree characteristics are given as maximum tree depth (*depth*), average tree depth (*depth_{avg}*), branching factor (*bf*), number of leaf nodes (*nLeaves*) and total number of tree nodes (*size*). Results in bold highlight typical benefits of adaptive BB-trees for a given bs.*

speed-up (first two lines), the proposed 2-means++ initialization improves significantly *both* the construction speed and the tree characteristics, since it provides less unbalanced trees while considerably reducing the tree size.

3.3.2 Tree search

In order to investigate performances of both sided and symmetrized/mixed Bregman NN queries, we first carried out experiments of both exact and approximate NN queries wrt the KL divergence. Tab. 3.3 displays results of exact search on the Corel dataset, for different settings of the maximum branching factor. We noticed a significant computational speed-up with re-

BB-tree construction ($bs = 50$)					
method	*iter*	*depth*	$depth_{avg}$	*Leav.*	*speed-up*
2-means	10	53	28.57	594	1
2-means++	10	58.33	31.18	647	1.03
2-means++	**0**	**20**	**10.76**	**362**	**19.71**

Table 3.2: *Comparison between different partitioning methods when building a BB-tree on the SIFT dataset. Bregman* `2-means++` *with iter* $= 0$ *means that only the initialization step is carried out. The speed-up is by comparison with the more computationally expensive k-means method.*

		leaves		
BF	speed-up	visited	total	ratio
2	2.11	725	2271	31.9%
3	2.22	731	2605	28.1%
4	2.21	752	2818	26.7%
10	2.23	740	2853	25.9%

Table 3.3: *Exact search statistics on the Corel dataset ($bs = 50$).*

spect to brute-force search because pruning conditions are frequently met, thus enabling to ignore a large proportion of leaves (as shown in the last column). Although overall results on *exact* search do not particularly improve those obtained by Cayton [Cay08], we interestingly note that relaxing the constraint on branching factor generally provides better results (*e.g.*, compare the speed-up for $BF = 2$ and $BF = 10$).

Rather than thoroughly investigating exact NN search, we mostly focused on approximate search, which turns out to be much more interesting for practical applications. Indeed, when performing queries on large databases (say, containing up to million descriptors), the nearest neighbor retrieval task can be generally relaxed to find a "good" NN, *i.e.*, a point that is close enough to the true NN. This approximation allows one to get much faster retrieval times. Namely, we used the maximum number of explored leaves as a parameter for stopping the branch-and-bound search. In each experiment of approximate search, we fixed a value of this parameter (from *near-exact* search to visiting only a *single* leaf), then we evaluated error rate and computational cost. The error rate is expressed as the number of closer points to the approximated NN (a quantity called *Number Closer*), while the speed-up is given by the ratio between the number of divergence computations in BB-tree over the brute-force search. Results on the two datasets are shown in Fig. 3.7 and 3.9 for different settings of construction parameters. Note that results are depicted in logarithmic scales (base 10), as they cover a large range of both speed-up and Number Closer values. Results on the SIFT dataset are particularly interesting, as they reveal impressive performance of BB-tree++; *e.g.*, the point marked by an asterisk in Fig. 3.7 refers to a speed-up of about 66 ($10^{1.8}$) for an error of 35.5 ($10^{1.55}$), *i.e.*, only 0.36% of overall database points. The improvement over Cayton's results [Cay08] is shown in Fig. 3.8. We gain up to a magnitude order over BB-tree when the Number Closer equals 32 ($10^{1.5}$), which corresponds to a relative error of 0.32% when compared with the overall number of database points. This average error can be considered neglectable in most NN retrieval applications. At this point, we reach a speed-up of 66 ($10^{1.8}$) against 1.7 ($10^{1.22}$) of Cayton (these two points are marked by asterisks in the figure). In Fig. 3.9, the marked point shows that a speed-up of 525 ($10^{2.72}$) corresponding to an

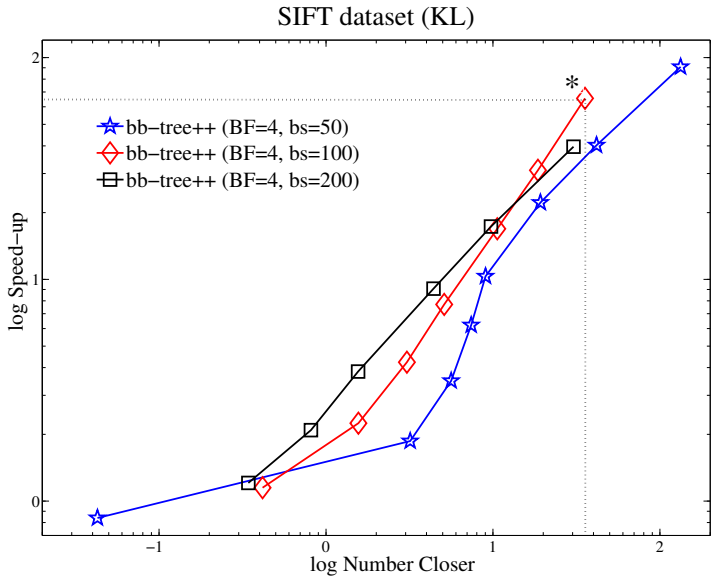


Figure 3.7: *Results of approximate NN retrieval on the SIFT dataset wrt. KL divergence (log-log plot).*

error of about 21 points ($10^{1.33}$), *i.e.* less than 0.4% of database points.

One of the main contributions of this chapter is to adapt the BB-tree data structure to perform *symmetrized* and *mixed* Bregman queries, as explained in Section 3.2.2. We used the SIFT dataset to test the symmetrized Bregman ball tree, where data are to be first converted to multinomial distribution parameters for computing symmetrized Bregman ball centroids, following the geodesic bisection algorithm of [NN09a]. As shown in Fig. 3.10, results are similar to those of sided Bregman queries (Fig. 3.7), with significant speed-ups for small error values (See Fig. 3.11 for a comparison with randomized sampling.) We also tested mixed Bregman queries on the Corel database (Fig. 3.12 and 3.13), which gave slightly better results than asymmetric queries. (Note the marked point in Fig. 3.12, where a speed-up of 205 ($10^{2.3}$) is reached when the average error is only 13 ($10^{1.1}$), corresponding to 0.02% of total database points.)

3.3.3 Bvp-tree construction

The low construction cost is a major advantage of Bvp-tree over other data structures like BB-trees, which usually require running Bregman $k-$means clustering [BMDG05]. Indeed, when using random vantage points, only one divergence has to be computed for each point in a Bregman ball. Hence, the overall construction time is $\mathcal{O}(n\delta)$, n being the dataset size and δ the

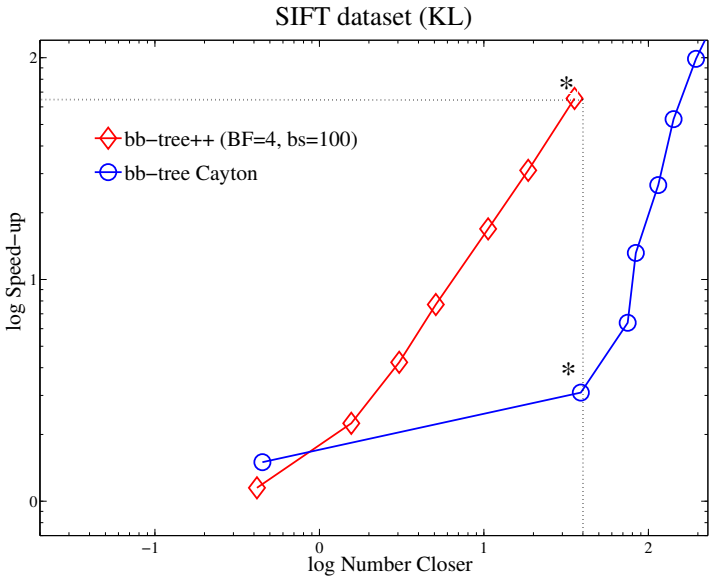


Figure 3.8: *Comparison of bb-tree and bb-tree++ for approximate search on the SIFT dataset (log-log plot).*

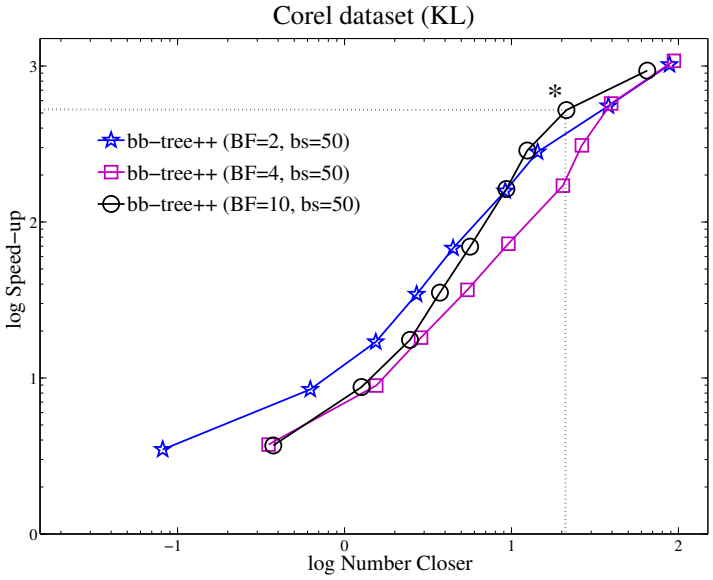


Figure 3.9: *Results of approximate NN retrieval on the Corel dataset wrt. KL divergence (log-log plot).*

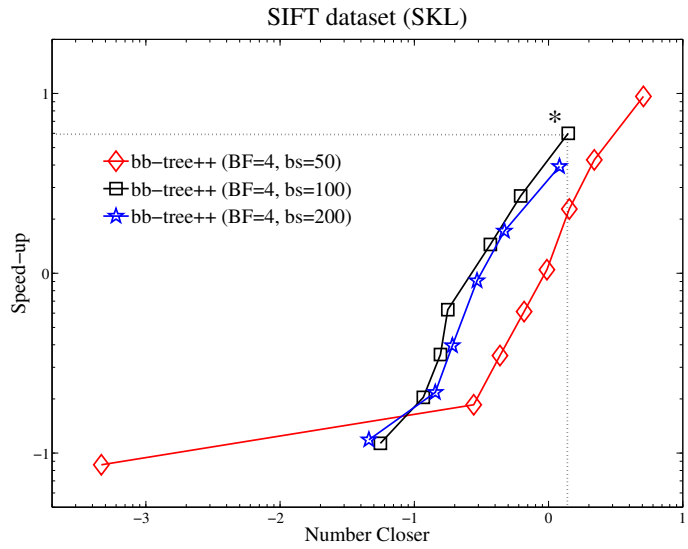


Figure 3.10: *Approximate search for symmetrized Bregman queries on SIFT dataset wrt. SKL divergence. The marked point shows a speed-up of* 60.3 ($10^{1.78}$) *when the Number Closer equals* 14 ($10^{1.15}$), i.e. 0.14% *of database points.*

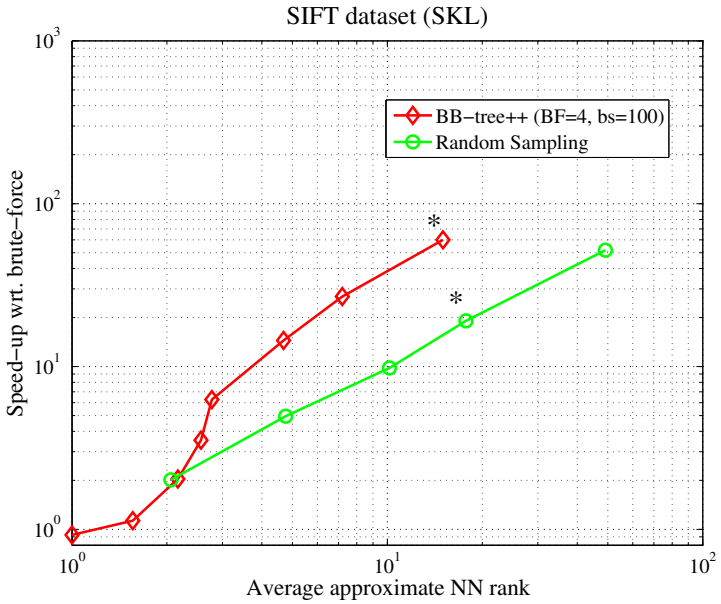


Figure 3.11: *Approximate search for symmetrized Bregman queries on SIFT dataset wrt. SKL divergence. The marked points show a speed-up of* 60.3 ($10^{1.78}$) *for BB-tree against* 19.1 ($10^{1.28}$) *of Random Sampling.*

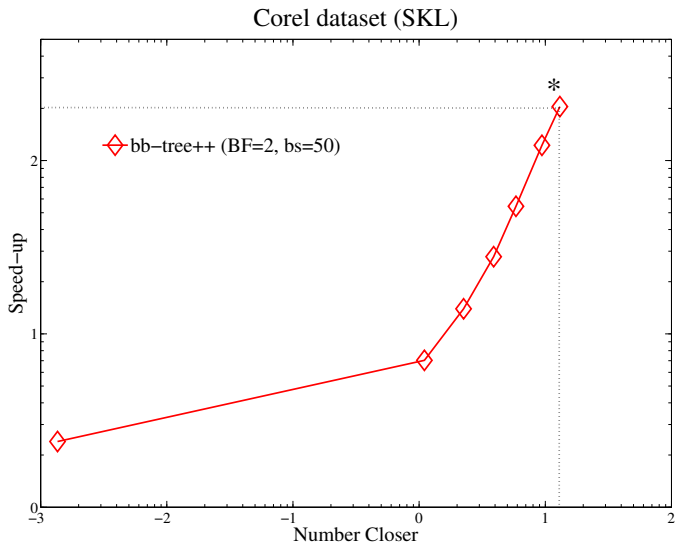


Figure 3.12: *Approximate search for mixed Bregman queries on Corel dataset* (i.e., *wrt. SKL divergence).*

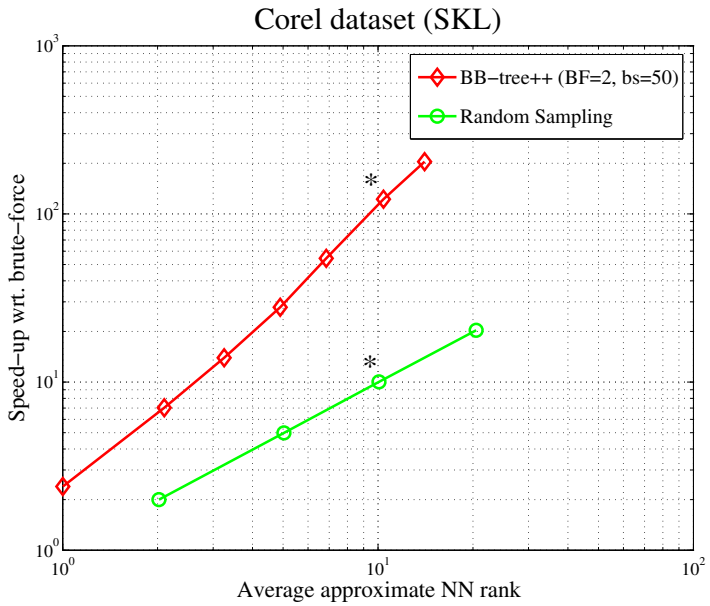


Figure 3.13: *Approximate search for mixed Bregman queries on Corel dataset wrt. SKL divergence. Marked points show that BB-tree++ outperforms randomized search up to an order of magnitude.*

dataset	*bs*	*depth*	$depth_{avg}$	*nLeaves*	*cost*
Corel	50	11	11	2048	$6.6 \cdot 10^5$
Corel	100	10	10	1024	$6.0 \cdot 10^5$
Corel	200	9	9	512	$5.4 \cdot 10^5$
dataset	r_0	*depth*	$depth_{avg}$	*nLeaves*	*cost*
Corel	$1.0 \cdot 10^0$	13	10.50	3790	$6.3 \cdot 10^5$
Corel	$1.5 \cdot 10^0$	13	10.09	3586	$6.1 \cdot 10^5$
Corel	$2.0 \cdot 10^0$	13	6.71	718	$4.0 \cdot 10^5$

Table 3.4: *Bvp-tree construction results for different stopping criteria: bucket size (bs) or leaf radius (r_0).*

average depth of the tree. (Note that the computational cost of BB-tree construction is $\mathcal{O}(rn\delta)$, with $r \geq 2$ increasing with the number of $k-$means iterations.) As mentioned in Sec. 3.2.3, we used either the *bucket size* (*bs*) or the *leaf radius* (r_0) criterion for stopping the node branching. The first criterion enables us to build a perfectly balanced vantage point tree, i.e., all leaves have equal depth and store the same number of data points. On the contrary, the second criterion enables to partition even large Bregman balls that contain very sparse data, thus giving a smaller number of leaves. Tab. 3.4 summarizes the typical characteristics of Bvp-trees for the Corel dataset, as well as the construction cost, for different values of the construction parameters. The construction cost is expressed as the total number of divergences computed when building the tree. The top half-table shows characteristics of trees built with the bucket size criterion. In this case, trees are perfectly balanced and the average tree depth equals the maximum one. The bottom half-table refers to trees built with the leaf radius criterion, which gives unbalanced trees, as proven by the difference between maximum and average tree depth. Also note that trees having roughly equal average depth, but constructed with different criteria, significantly differ in terms of the overall size (*i.e.*, number of leaves).

3.3.4 Bvp-tree search

We tested our Bvp-tree algorithm for both *exact* and *approximate* NN retrieval. Furthermore, we evaluated performances of both sided and symmetrized Bregman NN queries on the two datasets. Table 3.5 displays results of exact search for different settings of the tree construction. Performances are expressed in terms of the computational cost ratio between brute-force and Bvp-tree search. These results show a significant *order of magnitude* speed-up over the brute-force method for the Corel dataset. On the contrary, the SIFT dataset revealed to be more challenging because of the very high dimensionality of data. Interestingly, SKL queries showed

div	*dataset*	*bs=50*	*bs=100*	*bs=200*	Bb-tree
KL	Corel	2.12	**2.33**	2.04	2.4
KL	SIFT	0.90	0.95	**0.97**	0.9
SKL	Corel	**3.24**	3.13	2.79	-
SKL	SIFT	0.96	1.01	**1.05**	-

Table 3.5: *Performances of exact Bvp-tree search, measured as computational speed-up over brute-force search. The last column displays results reported by Cayton [Cay08] for Bb-trees.*

better performances than asymmetric KL queries. A comparison with results reported by Cayton [Cay08] for the KL divergence shows that Bvp-tree does not improve over BB-tree (last column in Table 3.5). However, experiments of SKL queries gave better results, although a direct comparison with BB-tree is not possible, as this latter cannot handle symmetrized queries.

Besides investigating exact search, we also tested our Bvp-tree for approximate NN retrieval. This approximation allows for significant speed-ups when searching the tree. In order to find an approximate NN, we performed the iterative Bvp-tree search procedure up to a maximum prescribed number of visited leaves. In each experiment, we fixed a value of this parameter, ranging from *near-exact* search to the exploration of a *single node*, then we evaluated the *Number Closer* (i.e., the number of closer points to the approximated NN) and the *Speed-up* (i.e., the ratio between the number of divergence computations of brute-force and tree search). Fig. 3.14 and 3.15 display the most significant results of such experiments (log-log plot).

§ 3.4 Conclusion

In this chapter, we have addressed the problem of performing efficient k-NN search when using a broad class of information-theoretic non-metric distances, *i.e.*, Bregman divergences. We have described generalizations of two existing data structures to this class of distorsion measures: Bregman Ball trees, that we improved in terms of computational efficiency, and Bregman Vantage Point trees, that we brought into the Bregman framework. Furthermore, we adapted these data structures in order to deal with symmetrized divergences as well, as these latter are often preferred for image retrieval and classification. Experimental results show that our work improves over the state-of-the-art, thus confirming our data structures as suitable techniques for efficient data indexing when Bregman divergences, like the Kullback-Leibler divergence, are expected to be more appropriate.

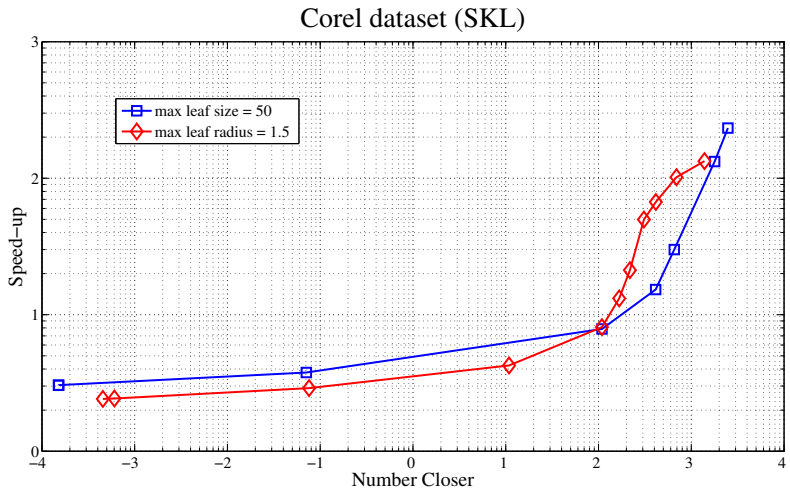


Figure 3.14: *Results of Bvp-tree approximate NN retrieval (log-log plot) on the Corel database.*

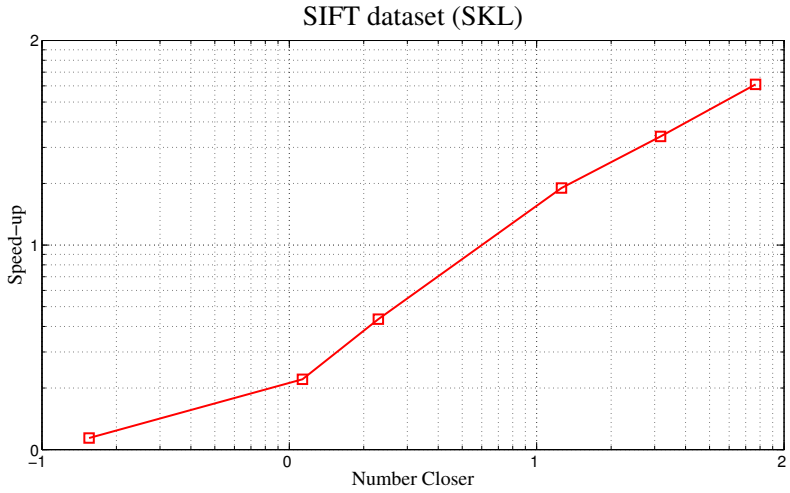


Figure 3.15: *Results of Bvp-tree approximate NN retrieval (log-log plot) on the SIFT database.*

PART II

INDUCING *k*-NEAREST NEIGHBOR CLASSIFIERS IN A BOOSTING FRAMEWORK

- CHAPTER 4 -

UNN: UNIVERSAL NEAREST NEIGHBORS

The k-nearest neighbors (k-NN) rule is one of the oldest and simplest methods for pattern classification [CH67]. Besides still being a core tool in certain machine learning domains, it has proven extremely successful in many *computer vision* applications, *e.g.*, object recognition [BMP02]. In particular, often image categorization relies on uniform voting among the nearest annotated instances (*prototypes*) in the space of descriptors. For example, this is the case for several methods aiming at matching local descriptors extracted from images, like SIFT descriptor matching [MS05]. Philosophically, k-NN techniques can be related to the theory of prototype-based categorization, as it has been developed by Rosch et al in the '70s [Ros73, RMG+76]. According to Rosch's theory, the way people categorize objects relies on matching them against the *prototype*, *i.e.*, an "ideal exemplar", which contains the most representative features inside the category.

On an empirical point of view, despite its simplicity, k-NN is still a very attractive tool to practitioners, as it generally shows very good performances in practical applications. Moreover, k-NN is naturally adapted to multi-class problems, where other more sophisticated tools, like SVM, fail to provide reasonable trade-off between performances and computational cost when dealing with a large number of classes.

However, in spite of the good generalization properties of the classic k-NN rule, it suffers from high variance when dealing with sparse prototype datasets in high dimension. A few techniques have been proposed with the aim of improving the k-NN classification, which generally rely either on deforming the nearest neighborhood relationship by learning the "most" appropriate distance function or modifying the input space by projecting data into the "best" low-dimensional subspace.

In this chapter, we briefly discuss such methods and describe a novel approach to generalize and improve the k-NN rule by tackling the problem at its core. In particular, we provide a new k-NN boosting algorithm,

that we call UNN (Universal Nearest Neighbors), for the *induction* of *leveraged* k-NN. Our approach consists in redefining the voting rule as a strong classifier that linearly combines predictions from the k closest prototypes. Therefore, the k-nearest neighbor examples act as weak classifiers and their weights, called *leveraging coefficients*, are learned in such a way to minimize a *surrogate risk*, which upper bounds the empirical misclassification rate over training data. Interestingly, this surrogate risk can be arbitrarily chosen from a class of *Bregman loss* functions, including the familiar exponential, logistic and squared loss. Unlike several existing approaches, our UNN algorithm does not require to learn a metric distance nor modify the input space. Indeed, UNN does not affect the k-nearest neighborhood relationship, but rather acts on top of k-NN search. Moreover, a major feature of UNN is the ability to learn which prototypes are the most relevant for a given class, thus allowing one for effective data reduction by filtering the training data.

Experimental results on the synthetic dataset of Ripley show that such a filtering strategy is able to reject "noisy" prototypes, and yields classification error close to the optimal Bayes error. Furthermore, the improvement of UNN over a state-of-the-art metric learning algorithm is shown by experiments on some datasets from the UCI repository. Finally, we concentrated on the image categorization task and carried out image categorization experiments on two databases containing eight and thirteen classes of natural scenes, respectively. In these experiments, we tested two different state-of-the-art image descriptors, Gist and Bag-of-Features. We show that our method outperforms the classic k-NN classification significantly, while enabling significant reduction of the computational cost thanks to data filtering. Furthermore, UNN compares favourably to other learning techniques on Bag-of-Features data.

§ 4.1 INTRODUCTION

4.1.1 Generic visual categorization

In this thesis, we address the problem of *generic visual categorization*. This is a relevant task in computer vision, which aims at automatically classifying images of real-world scenes into a discrete set of categories, such as *indoor* vs *outdoor* [PS05, GPP+07], *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

(a) (b)

Figure 4.1: *Two examples of common challenges involved in the image categorization task (8-cat scene database [OT01]): (a) high* intra-class *variability for category "tall buildings"; (b) low* inter-class *variability between categories "inside city" (left) and "street" (right).*

find both the presence and the position of a target object in an image, e.g. person detection).

Automatic categorization of generic real-world 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 *intra-class variability* (*i.e.*, visually different images may belong to the same category, like in Fig. 4.1(b)) and low *inter-class variability* (*i.e.*, distinct categories may contain visually similar images, like in Fig. 4.1(a)).

Classifying images requires an effective and reliable description of the image content, *e.g.*, 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 [SSL04, VS07], 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. In particular, the best classification performances are generally provided by methods combining the visual relevance of such low-level image descriptors with effective (supervised or unsupervised) learning techniques. For instance, Bosch et al [BZM08] have recently shown the effectiveness of a hybrid (generative/discriminative) learning approach to discover the occurrences of some "latent topics" in an image, using vector quantized local features. For an extensive survey of the most common scene classification techniques relying on such descriptors, see the paper of Bosch et al [BMM07].

4.1.2 *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 prop-

erties [SDI06]. A generalization of the k-NN rule to the *multi-label* classification framework has been also proposed recently by Zhang and Zhou [ZZ07], whose technique is based on the *maximum-a-posteriori* principle applied to multi-labeled k-nearest neighbors. 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, k-NN classification provably tends to the Bayes optimum when increasing the sample size.

Although such advantages make k-NN classification very attractive to practitioners, it is an algorithmic challenge to speed-up k-NN queries and design schemes that scale-up well with high dimensional datasets [JDS10]. Furthermore, reducing the misclassification rate of the k-NN rule 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 [DKJ+07]. 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 [Har68]. Furthermore, in a number of prior work, the classification problem has been reduced to tracking ill-defined categories of neighbors, interpreted as "noisy" [BM02].

Most of these recent techniques are in fact partial solutions to 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* [HA03]. This problem has been reformulated by Marin et al [MRT09] 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, say by decision stumps).

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*);
- *weighting* nearest neighbors;
- *boosting* nearest neighbors.

4.1.2.1 Learning the distance metric

Most approaches for improving the *k*-NN classification rule rely on learning an appropriate adaptive distance metric from training data. For instance, refer to the seminal work of Fukunaga and Flick [FF84], who presented an optimal global metric for *k*-NN. An analogous approach was later adopted by Hastie and Tibshirani [HT96], who carried out linear discriminant analysis to adaptively deform the distance metric. Then, Paredes [Par06] proposed a method for learning a weighted distance, where weights could be either global (*i.e.*, only depending on classes and features) or local (*i.e.*, depending on each individual prototype as well).

Several other metric learning techniques have been also proposed, as described in a recent survey [YJ06]. Often, such methods require to learn a *Mahalanobis distance* metric, *i.e.*, a linear transformation of the input feature space. This is accomplished by optimizing an objective function defined on either labelled training data or positive (same class) and negative (different class) training pairs. The main difference among these methods lies in the objective functions, which are usually tailored to the specific task at hand. For instance, Goldberger et al [GRHS04] have proposed a completely supervised algorithm, Neighborhood Component Analysis (NCA), that aims at learning the Mahalanobis distance metric that minimizes the expected *leave-one-out* test error from a stochastic variant of *k*-NN classification. Among the existing techniques for metric learning, Large Margin Nearest Neighbor (LMNN) [WS09] and Information Theoretic Metric Learning (ITML) are currently state-of-the-art. The first algorithm (LMNN) tries to improve the classification accuracy of the *k*-NN rule by constraining most of the data in a *k*-nearest neighborhood to have the same label. For this purpose, LMNN learns a linear transformation of the input space by minimizing a loss function that penalizes large distances between neighbors in the same class, while also penalizing small distances between examples of different classes. In this sense, the Mahalanobis metric, which is obtained as the solution to a convex semidefinite program, is to "deformate" the Euclidean (circular) neighborhood to an ellipsoid that maximizes the margin between the *k*-nearest neighbors and the rest of training data. (See Fig. 4.2.) The LMNN algorithm has been also successfully extended to the computer vision domain by Kumar et al [KTZ07], who have demonstrated its effectiveness for the face recognition task. The second state-of-the-art technique to metric learning for *k*-NN is ITML [DKJ+07], which is an information-theoretic approach to optimize the metric under a wide range of possible constraints and prior knowledge on the Mahalanobis distance. An objective function is defined, that controls the trade-off between satisfying the constraints on some positive and negative pairs, and regularization, which constraints the solution to be as close as possible to a given prior.

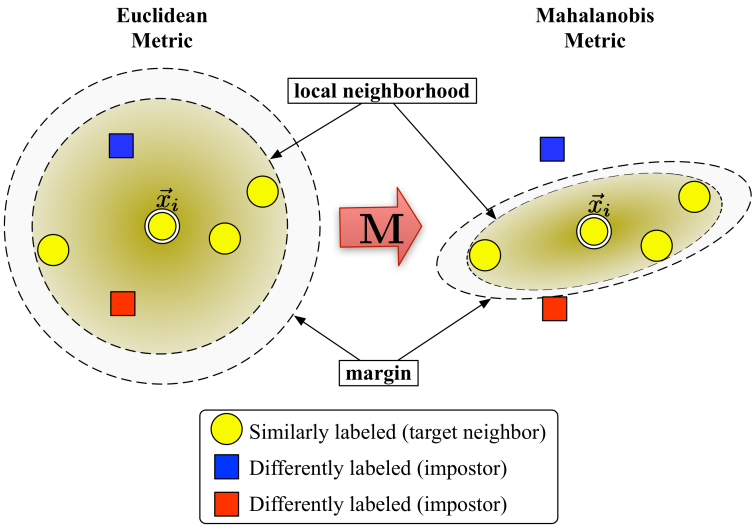


Figure 4.2: *Schematic illustration of the LMNN metric learning method [WS09].*

Finally, Guillaumin et al [GVS09] have recently proposed a new method for learning Mahalanobis metrics, whose effectiveness has been shown particularly for face identification. This method, called Logistic Discriminant Metric Learning (LDML), uses logistic discriminant to learn a metric from a set of labelled image pairs, by constraining positive pairs to have smaller distances than negative pairs.

4.1.2.2 Kernel nearest neighbors

Another class of techniques apply the nearest neighbors rule to data embedded in a high-dimensional feature space, according to the *kernel trick* approach of *support vector machines* (SVM). For example, Yu et al [YJZ02] 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, an effective technique, SVM-*k*NN, has been proposed by Zhang et al [ZBMM06], which only involves a "refinement" step at classification time, without requiring to learn the distance metric explicitely. 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 instances.

4.1.2.3 Weighted *k*-NN

A third class of *k*-NN methods relies on weighting nearest neighbors votes based on their distances to the query sample [Dud76]. Recently, Zuo et al [ZZW08] 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, a weighted k-nearest neighbor voting scheme has been tested as a *soft assignement* strategy for matching local descriptors in the context of object retrieval [PCI+08]. In particular, this strategy relies on the Radial Basis Function (RBF) in order to weight the contributions of different neighbors according to their distance from the query vector.

4.1.2.4 Boosting k-NN

Finally, only very few work have proposed the use of boosting techniques for k-NN classification. For instance, Amores et al [ASR06] use AdaBoost for learning a distance function to be used for k-NN search. On the other hand, García and Ortiz [GPOB09] 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 their method, depending on the way the input space is modified. Their first algorithm is based on optimal subspace selection, i.e., 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.

4.1.3 Overview of the chapter

In this chapter, we propose a complete solution to the problem of boosting k-NN classifiers. Namely, we propose the *first* boosting algorithm, called UNN, which *induces* a *leveraged* nearest neighbor rule that generalizes the uniform k-NN rule. Indeed, the voting rule is redefined as a strong classifier that linearly combines weak classifiers induced by the k-NN rule. Therefore, our approach does not need to learn a distance function, as it directly operates on the top of k-nearest neighbors 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 UNN boosting algorithm is an iterative procedure that learns the weights of weak classifiers, called *leveraging coefficients*. We show that this algorithm converges to the *global* minimum of any chosen *classification calibrated surrogate*[1] [BJM06]. Hence, our framework handles most popular losses in the

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

machine learning literature: squared loss, exponential loss, logistic loss, etc. In particular, we prove a specific convergence rate for the exponential loss (reported in our experiments) far better than the general rate of Nock and Nielsen [NN09c]. Another important characteristic of UNN is that it is able to discriminate the most relevant prototypes for a given class, thus allowing one for significant data reduction while improving at the same time classification performances.

In the following sections we present our approach to k-NN boosting. Sections 4.2.1-4.2.3 present key definitions for k-NN boosting. These sections also describe how to replace the classic uniform k-NN rule by a *leveraged* k-NN rule. Leveraged k-NN classifiers are induced by UNN algorithm, which is detailed in Sec. 4.2.4 for the case of the exponential risk. Sec. 4.2.5 presents the generic convergence theorem of UNN and the upper bound performance for the exponential risk minimization. Our experiments on both synthetic and image categorization datasets are reported in Sec. 4.3. Then, Sec. 4.4 discusses results and mentions future work. Finally, we postpone the general form of UNN algorithm and proofsketches of our theorems to Appendix A.

§ 4.2 METHOD

4.2.1 Problem statement and notation

In this work, we address the task of *multi-class*, *single-label* classification. Although the multi-label framework is quite well established in literature [BLSB04], 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. E.g., scene categorization with up to 899 real-world categories has been recently studied [XHE+10]. We treat the multi-class problem as multiple binary classification problems (*one-versus-all*) as it is customary in machine learning. Hence, for each class c, a query image is classified either 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.

When considering the image categorization task, the representation space is formed by extracting 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 membership of $\boldsymbol{x}$. In particular, the sign of component 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 Zhu et al [ZRZH09], we constrain class vectors to be *symmetric*, that is:

$$\sum_{c=1}^{C} y_c = 0 \,. \tag{4.1}$$

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

$$y_{\tilde{c}} = 1, \quad y_{c \neq \tilde{c}} = -\frac{1}{C-1} \quad (c = 1, 2, \ldots, C) \,. \tag{4.2}$$

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 refer to an input set of m examples $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m\}$, *e.g.*, arising from annotated images, which form the *training set*.

4.2.2 Boosting k-NN for minimization of surrogate risks

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, *i.e.*, to predict their true labels. Therefore, we aim at determining a classification rule $\boldsymbol{h}$ from the example dataset, which is able to minimize the classification error over all possible new observations. But 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 $\mathcal{S}$ without overfitting. This corresponds to defining a classifier that correctly classifies most of the example data themselves, thus minimizing the classification error over the example dataset (*empirical* or "true" *classification loss*). Therefore, 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:

$$\varepsilon^{0/1}(\boldsymbol{h}, \mathcal{S}) \doteq \frac{1}{C} \sum_{c=1}^{C} \frac{1}{m} \sum_{i=1}^{m} \left[\varrho(\boldsymbol{h}, i, c) < 0\right] \,, \tag{4.3}$$

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

$$\varrho(\boldsymbol{h}, i, c) \quad \doteq \quad y_{ic} h_c(\boldsymbol{x}_i) \tag{4.4}$$

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 the edge is positive (resp. negative) when the membership predicted by classifier and the actual example's membership agree (resp. disagree). Therefore, the empirical loss (4.3) 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 that 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. Unfortunately, minimizing the empirical risk is mathematically 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 (4.3) by that of a so-called *surrogate risk* [BJM06], to make the optimization problem amenable. In boosting, it amounts to sum (or average) over classes and examples a real-valued function ψ, called *surrogate loss*, thus ending up with the following rewriting of (4.3):

$$\varepsilon^{\psi}(\boldsymbol{h}, \mathcal{S}) \doteq \frac{1}{C} \sum_{c=1}^{C} \frac{1}{m} \sum_{i=1}^{m} \psi(\varrho(\boldsymbol{h}, i, c)) \ . \tag{4.5}$$

Common choices available for ψ include:

$$\psi^{\text{sqr}} \doteq (1 - x)^2 \ , \tag{4.6}$$
$$\psi^{\text{exp}} \doteq \exp(-x) \ , \tag{4.7}$$
$$\psi^{\text{log}} \doteq \log(1 + \exp(-x)) \ ; \tag{4.8}$$

(4.6) is the squared loss [BJM06], (4.7) is the exponential loss [SS99], and (4.8) is the logistic loss [BJM06].

Surrogates play a fundamental role in supervised learning. They 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 ($\varrho(\boldsymbol{h}, i, c)) > 0$) with large "confidence" values ($|\varrho(\boldsymbol{h}, i, c)) > 0| \gg 0$). Therefore, the minimization of surrogates remarkably impacts on that of the empirical risk, thus enabling to provide optimization algorithms with good generalization properties [NN09c].

In the following, we move from recent advances in boosting with surrogate risks to redefine the k-NN classification rule. In particular, we concentrate on the exponential risk and provide a new algorithm that learns a *leveraged* k-NN classifier, while provably converging to the *global optimum* of a surrogate risk. Our algorithm, called UNN (Universal Nearest Neighbors), meets boosting-type convergence properties under two mild

assumptions on the training set, *i.e.*, the properties of *weak learning* and *weak coverage*. In Appendix, we describe how our UNN algorithm generalizes to *any* surrogate loss, and provide the most general analysis, as well as the information-geometric interpretation of our surrogate minimization procedure.

4.2.3 Leveraged 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 $(\boldsymbol{x}, \boldsymbol{y})$ 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. The dissimilarity function thus may not necessarily satisfy the triangle inequality of metrics [2]. The k-nearest neighborhood relationship is intrinsically asymmetric, *i.e.*, $\boldsymbol{x_i} \in \mathrm{NN}_k(\boldsymbol{x})$ does not necessarily imply that $\boldsymbol{x} \in \mathrm{NN}_k(\boldsymbol{x_i})$. Indeed, a k-nearest neighbor of $\boldsymbol{x}$ does not necessarily contain $\boldsymbol{x}$ among its own k-nearest neighbors.

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

$$h_c(\boldsymbol{x}_q) = \sum_{j \,:\, \boldsymbol{x}_j \in \mathrm{NN}_k(\boldsymbol{x}_q)} \left[y_{jc} > 0 \right] \ , \tag{4.9}$$

where $\boldsymbol{x}_q$ is a test instance, h_c is the one-versus-all classifier for class c and square brackets denote the indicator function. Hence, the classic k-NN classification is based on majority vote among the k closest prototypes to a test observation.

We propose to weight the votes of nearest neighbors by means of real coefficients, thus generalizing (4.9) to the following *leveraged* k-NN rule $\boldsymbol{h}^\ell = \{h_c^\ell \ : \ c = 1, 2, ..., C\}$:

$$h_c^\ell(\boldsymbol{x}_q) = \sum_{j \,:\, \boldsymbol{x}_j \in \mathrm{NN}_k(\boldsymbol{x}_q)} \alpha_{jc} y_{jc} \ , \tag{4.10}$$

where $\alpha_{jc} \in \mathbb{R}$ is the leveraging coefficient for example j in class c, with $j = 1, 2, ..., m$ and $c = 1, 2, ..., C$. Hence, the classifier (4.10) linearly combines class labels of the k nearest neighbors (defined in Eq. 4.2) with their leveraging coefficients. The definition of our leveraged classification rule (4.10) can be conveniently rewritten as follows:

$$h_c^\ell(\boldsymbol{x}_q) = \sum_{t=1}^{T} \alpha_{tc} y_{tc} \left[\boldsymbol{x}_t \in \mathrm{NN}_k(\boldsymbol{x}_q) \right] \ , \tag{4.11}$$

[2] Although our method is general, all experimental results presented in the following sections refer to nearest neighbors with respect to the Euclidean distance.

i.e., as a weighted voting among $T \leq m$ prototypes selected from the training dataset $\mathcal{S}$, where only votes from the k-nearest neighbors are in fact non-zero valued. This redefinition is exactly equivalent to (4.10) when setting $T = m$; otherwise, it can be interpreted as an approximation of (4.10) due to filtering the prototype dataset, *i.e.*, searching for k-NN in a (possibly sparse) subset of the most relevant prototypes. This latter case is particularly interesting for our purpose, as UNN is actually able to considerably reduce the amount of prototypes needed for classification (see the following sections).

The main contribution of our work is to define a general algorithm (UNN) for learning the prototypes and their leveraging coefficients from training data. This algorithm operates on the top of classic k-NN methods, for it does not affect the k-nearest neighbor search when inducing weak classifiers of (4.10). Indeed, it is independent on the way nearest neighbors are computed, unlike most of the approaches mentioned in Sec. 4.1.2, which rely on modifying the neighborhood relationship via metric distance learning or kernel transformations. Indeed, our approach is still fully compatible with any underlying (metric) distance and data structure for k-NN search, as well as possible kernel transformations of the input space.

4.2.3.1 Edge matrix

The key ingredient for bringing the boosting principle into the prototype-based (k-NN) classification rule is to highlight the role of labelled examples as prototypes that (possibly) vote for each others. This is clear when interpreting (4.11) on training data themselves,

$$h_c^\ell(\boldsymbol{x}_i) = \sum_{j=1}^{m} \alpha_{jc} y_{jc} \left[\boldsymbol{x}_j \in \mathrm{NN}_k(\boldsymbol{x}_i)\right] \ , \tag{4.12}$$

that is, computing the classification score for $\boldsymbol{x}_i$ while searching for its k-nearest neighbors among all m data in $\mathcal{S}$. Because $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ vary in the same dataset, a given training observation $\boldsymbol{x}_l$ $(l = 1, 2, \ldots, m)$ can be interpreted as:

- either a *test* vector for which the k-nearest neighbors vote according to rule (4.12) with $i = l$;
- or a *prototype* vector that may possibly vote for another annotated instance (considered as test vector). This amounts to fixing $j = l$ and make i vary over all possible indices between 1 and m. In particular, due to the argument of the indicator function (in square brackets) in (4.12), $\boldsymbol{x}_l$ can actually vote only for its *reciprocal nearest neighbors* (*i.e.*, those points for which $\boldsymbol{x}_l$ itself is one of the k-nearest neighbors).

This twofold role of training instances in our leveraged k-NN framework is depicted in Fig. 4.3, where point $\boldsymbol{x}_i$ is shown to be either test point with respect to its k-nearest neighbors (left), or prototype voting for its reciprocal k-NN (right). Notice that in general the two sets of k-NN and reciprocal k-NN for a given point are not the same. (Typically, the closer a point is to the dataset "centroid", the more reciprocal k-NN it has, and viceversa [RNI09].) The information about k-NN and reciprocal k-NN, *i.e.*, on training examples and their corresponding prototypes, is coded into the so-called k-NN *edge matrix* $\mathrm{R}^{(c)} \in \mathbb{R}^{m \times m}$, which is defined for each class $c = 1, 2, ..., C$ as follows:

$$\mathrm{r}_{ij}^{(c)} \doteq \begin{cases} y_{ic} y_{jc} & \text{if} \quad \boldsymbol{x}_j \in \mathrm{NN}_k(\boldsymbol{x}_i) \\ 0 & \text{otherwise} \end{cases} . \tag{4.13}$$

The name of $\mathrm{R}^{(c)}$ is justified by an immediate parallel with (4.4). Indeed, each example j serves as a prototype (*i.e.*, weak classifier) for each example $\boldsymbol{x}_i$, predicting 0 if $\boldsymbol{x}_j \notin \mathrm{NN}_k(\boldsymbol{x}_i)$, y_{jc} otherwise, for the membership to class c. Hence, the j^{th} column of matrix $\mathrm{R}^{(c)}$ collects all edges of "classifier" $\boldsymbol{x}_j$, whereas the i^{th} row collects edges of all classifiers for example $\boldsymbol{x}_i$. (Note that non-zero entries of the j^{th} column correspond to the reciprocal k-NN of $\boldsymbol{x}_j$, whereas non-zero entries of the i^{th} row correspond to the k-NN of $\boldsymbol{x}_i$.) By multiplying both sides of Eq. (4.12) by y_{ic}, we obtain the following expression for the edge of the leveraged k-NN rule on example i for class c is:

$$\varrho(\boldsymbol{h}^{\ell}, i, c) = \sum_{j=1}^{m} \alpha_{jc} \mathrm{r}_{ij}^{(c)} . \tag{4.14}$$

Eventually, the induction of the leveraged k-NN classifier $\boldsymbol{h}^{\ell}$ amounts to fitting all α_{jc} so as to minimize (4.5), after replacing the argument of $\psi(\cdot)$ in (4.5) by (4.14).

4.2.4 UNN: learning of classifier $\mathbf{h}^{\ell}$

We propose a novel classification algorithm which induces the leveraged nearest neighbors classifier $\boldsymbol{h}^{\ell}$ (Eq. 4.10). In this section, we explain UNN specialized for the exponential risk minimization, with pseudo-code shown in Alg. 3. However, our analysis is much more general, as it involves the broad class of classification-calibrated surrogate risks [BJM06], and is postponed to Appendix in order not to burden the methodology. Like common boosting algorithms, UNN operates on a set of weights w_i ($i = 1, 2, ..., m$) defined over training data. Such weights are repeatedly updated to fit all leveraging coefficients α_{jc} of (4.10). At each iteration, the index to leverage, $j \in \{1, 2, ..., m\}$, is obtained by a call to a *weak index chooser* oracle $\textsc{Wic}(\cdot, \cdot, \cdot)$, whose implementation is detailed later in this section. As for the notation,

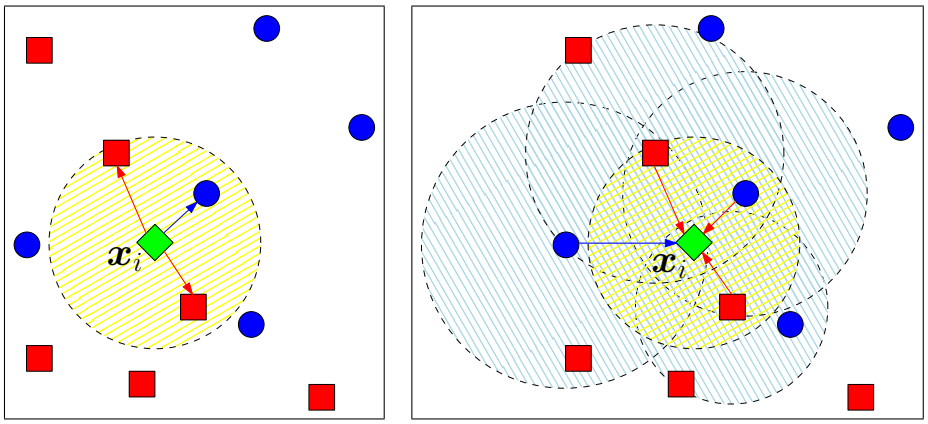


Figure 4.3: *Schematic illustration of the direct (left) and reciprocal (right) k-nearest neighbors ($k = 3$) of example $\boldsymbol{x}_i$ (green diamond). Red squares and blue circles represent examples of positive and negative classes. Each arrow connects an example to its k-nearest neighbors.*

remark that prototypes that contribute to the output classifier may be constrained to a reduced dataset $\mathcal{S}' \subseteq \mathcal{S}$, as our algorithm is able to perform effective data selection by training. Hence, we call the reduced example dataset "prototype" set and denote by NN'_k the k-NN computed on these data.

Watch Fig. 4.4 for a block diagram of UNN algorithm. In particular, notice that the initialization step, relying on k-NN search and edge matrix computation, is clearly distinguished from the iterative procedure, where at each iteration a new prototype is added, thus updating both the strong classifier $\boldsymbol{h}^\ell(\boldsymbol{x})$ and the weights w_i.

The training phase is implemented in a one-versus-all fashion, *i.e.* 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 $\bar{c}$, *i.e. any other class*. Eventually, one leverage coefficient (α_{jc}) per class is learned for each weak classifier (indexed by j). In the Appendix, we show that Alg. 3 is a specialization of a very general classification algorithm, thus justifying the name "Universal Nearest Neighbors". Namely, Alg. 3 induces the leveraged k-NN classifier by minimizing the exponential surrogate risk (4.7), very much like regular boosting does it for inducing a weighted voting rule for a set of weak classifiers.

The key observation when training weak classifiers using 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. (See Sec. 4.2.3.1.) 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

non-zero entries in matrix (4.13) on column j). This brings to a remarkable simplification when computing δ_j in step **[I.2]** and updating weights w_i in step **[I.3]** (Eq. 4.18, 4.19). Indeed, only weights of reciprocal nearest neighbors of j are involved in these computations, thus allowing us to store only the sparse not-null entries of matrix $\mathrm{R}^{(c)}, c = 1, 2, ..., C$. Notice that the set of reciprocal neighbors is splitted in two subsets, containing examples that agree (respectively, disagree) with the class membership of j, thus yielding the partial sums w_j^+ and w_j^- of (4.17).

Note that when whichever w_j^+ or w_j^- is zero, δ_j in (4.18) is not finite. There is however a simple alternative, inspired by [SS99], which consists in smoothing out δ_j when necessary, thus guaranteeing its finiteness without impairing convergence. More precisely, we suggest to replace:

$$w_j^+ \leftarrow w_j^+ + \frac{1}{m} , \tag{4.15}$$

$$w_j^- \leftarrow w_j^- + \frac{1}{m} . \tag{4.16}$$

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: $T = m, \textsc{Wic}(\{1, 2, ..., m\}, t, c) \doteq t$;

[b] the boosting approach: we first pick $T \geq m$, then we let j be chosen by $\textsc{Wic}(\{1, 2, ..., m\}, t, c)$ such that δ_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]**.

4.2.5 Properties of UNN

In this section, we enunciate two fundamental theorems for UNN. The first theorem reports a general monotonic convergence property of UNN to the optimal loss, for *any* given surrogate function. The second theorem further refines this general convergence theorem by providing effective convergence bound for the exponential loss. In particular, we constrain ψ to meet the following conditions:

(i) $\mathrm{im}(\psi) = \mathbb{R}_+$;

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

(iii) ψ is strictly convex and differentiable.

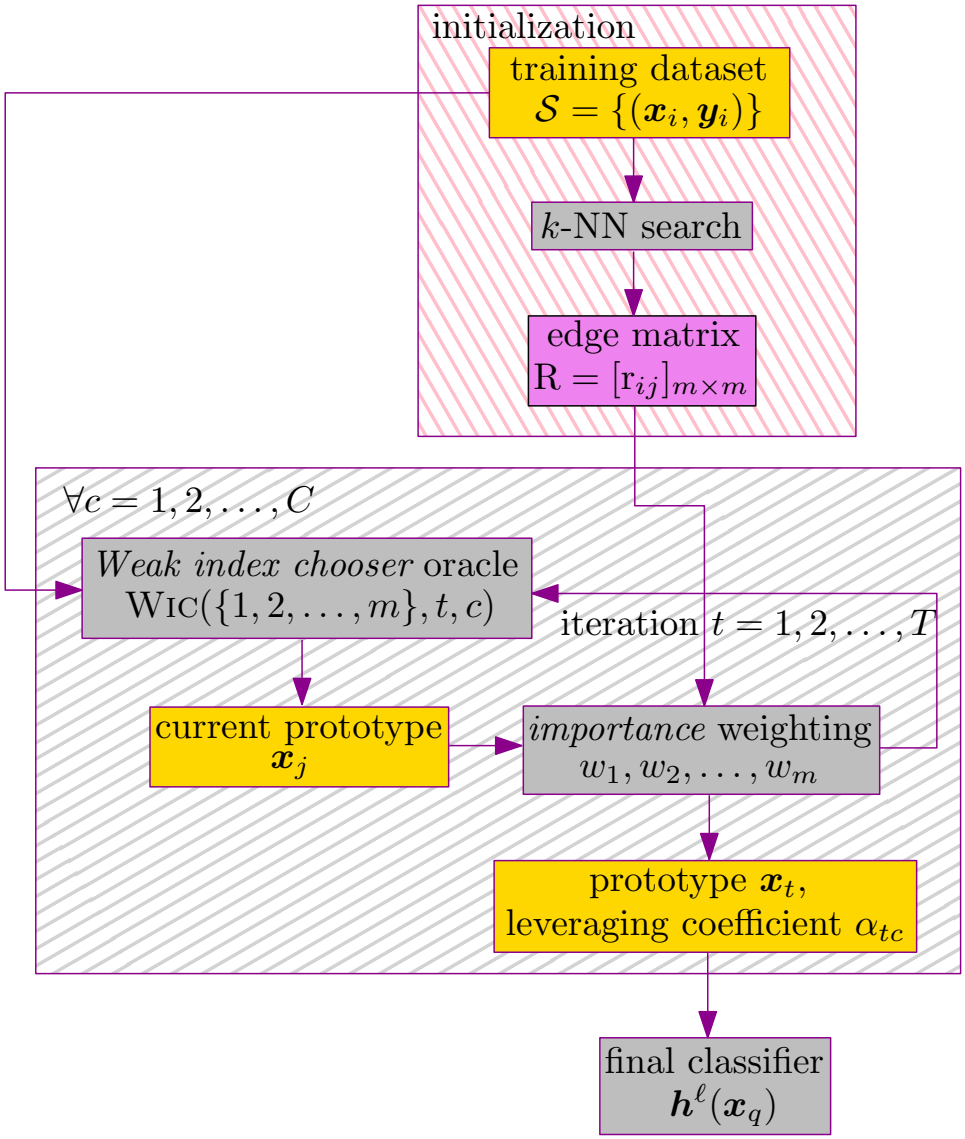


Figure 4.4: *Block diagram of the* UNN *learning scheme.*

Algorithm 3 UNIVERSAL NEAREST NEIGHBORS UNN($\mathcal{S}$) for $\psi = \psi^{\exp}$

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

Let $\mathrm{r}_{ij}^{(c)} \doteq \begin{cases} y_{ic} y_{jc} & \text{if } \boldsymbol{x}_j \in \mathrm{NN}_k(\boldsymbol{x}_i) \\ 0 & \text{otherwise} \end{cases} \quad \forall i, j = 1, \ldots, m,\ c = 1, \ldots, C;$

for $c = 1, 2, ..., C$ **do**

Let $\alpha_{jc} \leftarrow 0, \quad \forall j = 1, 2, ..., m$

Let $w_i \leftarrow 1, \quad \forall i = 1, 2, ..., m$

for $t = 1, 2, ..., T$ **do**

[I.1] Weak index chooser oracle: Let $j \leftarrow \mathrm{WIC}(\{1, 2, ..., m\}, t)$

[I.2] Let

$$w_j^+ = \sum_{i : \mathrm{r}_{ij}^{(c)} > 0} w_i, \quad w_j^- = \sum_{i : \mathrm{r}_{ij}^{(c)} < 0} w_i \,, \tag{4.17}$$

$$\delta_j \leftarrow \frac{1}{2} \log \left(\frac{w_j^+}{w_j^-} \right) ; \tag{4.18}$$

[I.3] Let

$$w_i \leftarrow w_i \exp(-\delta_j \mathrm{r}_{ij}^{(c)}), \quad \forall i \ : \ \boldsymbol{x}_j \in \mathrm{NN}_k(\boldsymbol{x}_i) \,; \tag{4.19}$$

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

Output: $h_c(\boldsymbol{x}_q) = \sum_{t=1}^T \alpha_{tc} y_{tc} \left[\boldsymbol{x}_t \in \mathrm{NN}'_k(\boldsymbol{x}_q) \right], \quad \forall c = 1, 2, \ldots, C$

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 [NN09b].

Theorem 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 (4.5), for any ψ meeting conditions **(i)**, **(ii)** and **(iii)** above.* (proofsketch in Appendix A)

Although we prove the boosting ability of UNN for all applicable surrogate losses, we choose to show in particular its behavior for the exponential loss $\psi^{\exp}$, which features far better convergence bound than the general one [NN09b].

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 [SS99]. (See Eq. (4.17) in Alg. 1 for the definition of w_j^+ and w_j^-.)

(WIA) let $p_j \doteq w_j^+ / (w_j^+ + w_j^-)$ and $||\boldsymbol{w}||_1 = \sum_{i=1}^m w_i$. There exist some $\gamma > 0$ and $\eta > 0$ such that the following two inequalities hold for

index j returned by $\textsc{Wic}(.,.,.)$:

$$|p_j - 1/2| \geq \gamma \ , \tag{4.20}$$
$$(w_j^+ + w_j^-)/||\boldsymbol{w}||_1 \geq \eta \ . \tag{4.21}$$

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

Inequality (4.20) is the usual weak learning assumption [SS99], when considering examples as weak classifiers. But a *weak coverage assumption* (4.21) is needed as well, because insufficient *coverage* of the reciprocal 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 $\textsc{Wic}(.,.,.)$, even when each one may be chosen more than once under the **WIA**. Last but not least, Theorem 4 also displays the fact that classification (4.20) may be more important than coverage (4.21).

4.2.5.1 Observations

The theorems proven above 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 [BJM06, NWJ09, YW10], 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 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 [BJM06]. 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 arbitrarily $C \geq 2$ in the single-label prediction problem. [BT07] proved an additional result for AdaBoost [SS99]: 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 [BT07]'s bound implies that we leverage a proportion of $1/m^{1-\eta}$ examples, "filtering out" the remaining ones. Moreover, the results of [NWJ09] 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 exist 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 [YW10] 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, [YW10] 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 [YW10] reveals that the reject loss is:

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

§ 4.3 EXPERIMENTS

In this section, we present experimental results of UNN on both synthetic and real datasets. Our experiments aim at carefully quantifying and explaining the gains brought by boosting on nearest neighbors voting. We first present results of UNN on two-class synthetic data (Sec. 4.3.1), which clearly enlighten the data reduction ability of our technique. Furthermore, we carried out experiments on some standard UCI datasets that are commonly employed for testing nearest neighbors-based techniques [AN]. We present a comparison on these datasets with both regular k-NN and a state-of-the-art metric learning method (Section 4.3.2). Then, we carried out experiments of multi-class scene categorization on two broadly used datasets of natural images. In Sec. 4.3.3 we discuss classification results of UNN compared to those of plain k-NN when using global Gist descriptors. Finally, in Sec. 4.3.4 we provide an extensive comparison of UNN with the

state-of-the-art categorization technique relying on the Bag-of-Features representation.

4.3.1 Synthetic datasets

In order to investigate the experimental behaviour of UNN we first tested our method on the synthetic Ripley's dataset [Rip94], which consists of two classes [3]. Each population of this dataset is an equal mixture of two two-dimensional normally distributed populations, which are equally likely. Training and testing datasets consist of 250 and 1000 points, respectively. (Testing data are displayed in Fig. 4.5.) Since the true underlying distributions are known, one can compute the optimal classification boundary of the Bayes rule (shown in Fig. 4.5 as well), which corresponds to the best theoretical error rate of 8.0% [Rip94].

Fig. 4.6 validates the monotonous decay of the exponential risk (4.7) on this dataset, as it is stated by Theorem 4 under the two basic weak index/learning assumptions (Sec. 4.2.5). Namely, we found out that the implementation of the WIC oracle, *i.e.*, the criterion for selecting a weak classifier at each iteration, significantly impacts on the convergence rate. Namely, we compared the "boosting" approach (*i.e.*, choosing the best weak classifier at each round) with two "lazy" implementations, where UNN only tests one weak classifier per iteration. (Examples are chosen either in random order or following a class-based order.) Notice that the boosting version of the WIC oracle provides much faster decay of the surrogate risk, thus outperforming the two alternative implementations. Indeed, the quick decrease in the first few iterations with boosting (red curve) corresponds to selecting the most discriminative examples, whereas the slow decrease in the next steps is due to adding more and more non relevant examples, thus overfitting the final classifier on the training data.

Classification results of UNN are shown in Fig. 4.7 as a function of the proportion θ of annotated examples that are retained for testing. The reported results were obtained for $k = 5$. They are compared to results of regular k-NN with random sampling, whose performances are averaged over a number of runs. Two main conclusions can be drawn from these results. First, UNN significantly outperforms the uniform voting for any value of θ, 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. 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 in UNN testing error between using all the examples and retaining the smallest subset.)

[3] Ripley's dataset is part of the matlab Statistical Pattern Recognition Toolbox, which is available at http://cmp.felk.cvut.cz/cmp/software/stprtool/.

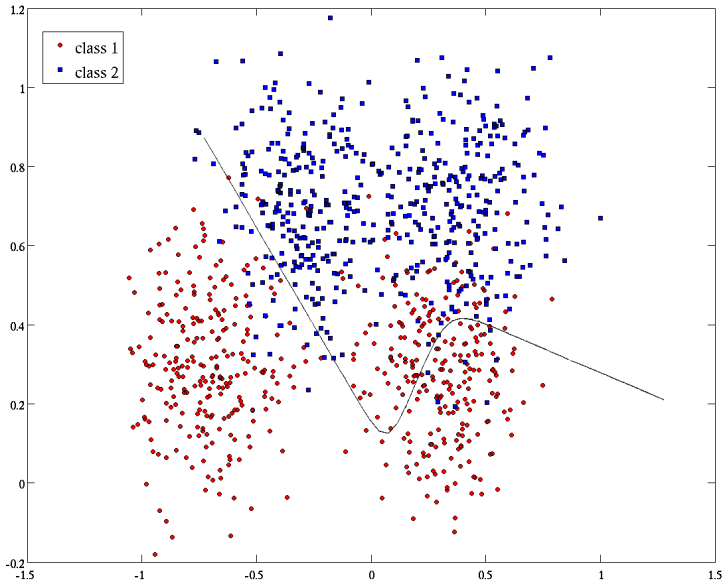


Figure 4.5: *Testing data for the Ripley's dataset. The theoretical Bayes boundary is also displayed as reported in [Rip94].*

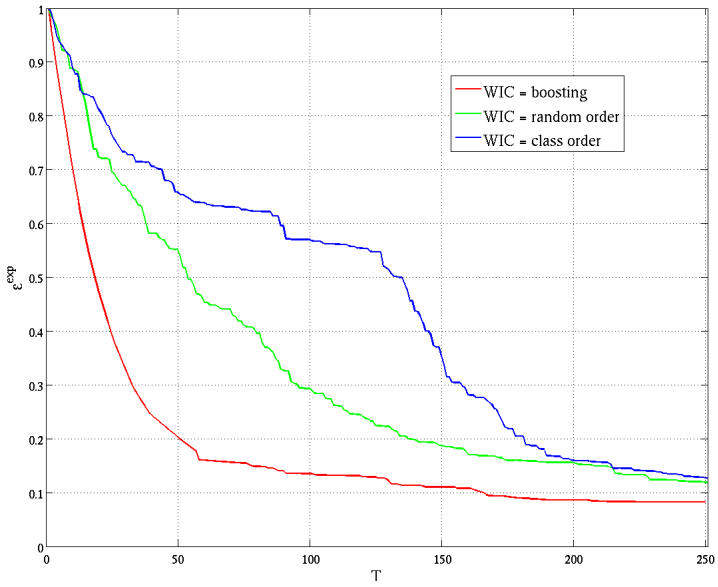


Figure 4.6: *Decrease of $\varepsilon^{\exp}(\boldsymbol{h}^{\ell}, \mathcal{S})$ as a function of the number of* UNN *iterations T for the Ripley's dataset with different oracle implementations. Notice that the boosting implementation (***[b]***, Sec. 4.2.4) always guarantees monotonic decrease of the surrogate loss, until the weak assumptions are matched (red curve). Conversely, the lazy implementation (***[a]***, Sec. 4.2.4) may select, at a given step, a classifier that does not match those assumptions, thus preventing the loss from strictly decreasing (see green and blue curves).*

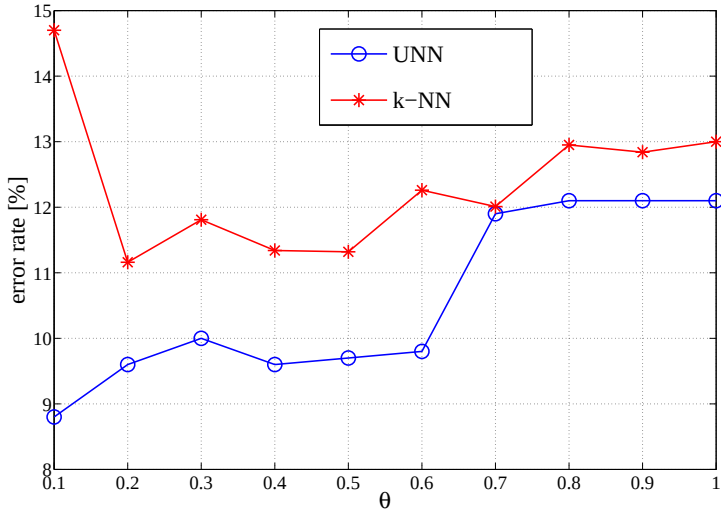
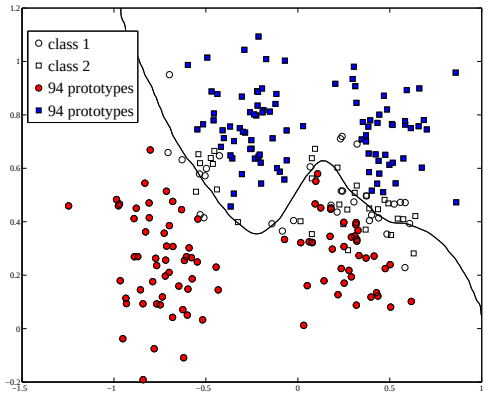


Figure 4.7: *Testing error for* UNN *and k-NN (random sampling) on the Ripley's dataset as a function of the proportion θ of examples retained for classification. Bayes rule (the optimal classifier) achieves 8% error.*

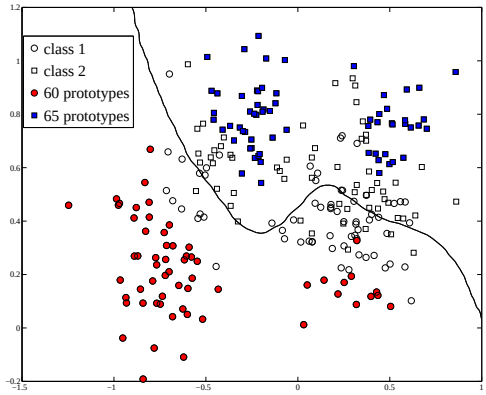
Indeed, assuming standard sampling assumptions [SFBL98], filtering actually benefits from two positive effects. The first is a *margin* effect, well known for *induced* classifiers [SFBL98]. The goodness-of-fit of the k-NN rule is driven by the most accurate examples, *i.e.* those surrounded by examples 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 A.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 4.8(c), makes this gap maximization translate 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. 4.8(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.

4.3.2 UCI datasets

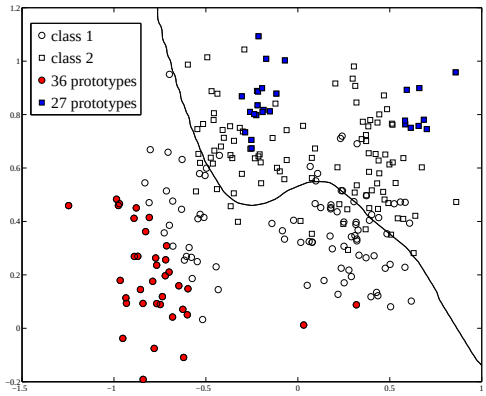
We carried out experiments using UNN on standard UCI datasets, and compared our method with both regular k-NN classification (with respect



(a)



(b)



(c)

Figure 4.8: *Maps of Ripley's training data and selected 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. For this reason, when* $\theta = 0.25$*, filtering produces a clear-cut gap between the two possible membership predictions (but not between the original classes). The resulting classification boundary between classes is shown as well. Interestingly, while this frontier still does not separate the original classes (without error), it does separate the memberships predictions, with much larger minimal margin. The combination of the data reduction and polarity reversal for memberships has thus simplified the learning of* $\mathcal{S}$*, and eased the capture of the optimal frontier with nearest neighbors.*

to the Euclidean distance) and a state-of-the-art metric learning algorithm, ITML, proposed by Davis et al [DKJ+07]. 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 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. 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 [DKJ+07], using the same settings as in their paper. We evaluated classification performances on five runs of two-fold cross-validation. In Table 4.1 we present average results over all cross-validation runs, as well as the values used for k. The same results are presented as histograms in Fig. 4.9, where the Binomial confidence intervals at the 95% level are shown as well. Globally, UNN outperforms both ITML and k-NN. In particular, UNN significantly outperforms regular k-NN classification (except for *"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 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 as reference instances for classification.)

4.3.3 Image Categorization using global Gist descriptors

We tested our UNN classification method on global descriptors for the categorization of natural images. In particular, we used the database of natural scenes collected by [OT01], which has been successfully used to validate several classification techniques relying on their Gist image descriptor. 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

Table 4.1: *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

related to the spatial organization of dominant scales and orientations in the image. Using such a global representation instead of a local one has the main advantage of providing a single compact descriptor of an image. In particular, the ability of mapping an image 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 explicitely 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 [GD05]. *E.g.*, this is the case for Bag-of-Features methods, that we discuss in Sec. 4.3.4 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 [TMFR03].

In the following, we denote as *8-cat* the database of [OT01], which contains 2,688 color images of outdoor scenes of size 256x256 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. 4.10. In addition, we carried out categorization experiments on a larger database of 13 categories as well, denoted as *13-cat*. This dataset was firstly proposed by [FFP05] and contains five more categories, as shown in Fig. 4.11. We extracted Gist descriptors from these images with the most common settings: 4 resolution levels of the Gabor pyramid, 8 orientations

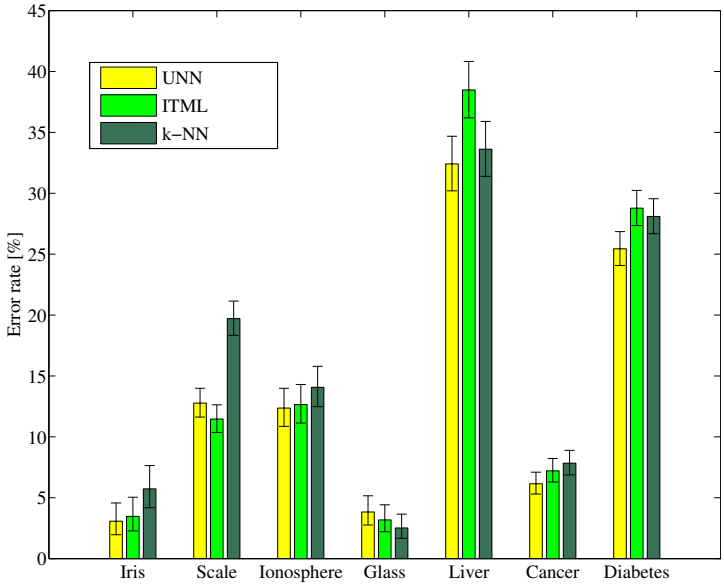


Figure 4.9: *Classification error rates for* UNN, ITML *and regular k-NN classification. The 95% confidence intervals are also shown.*

per scale and 4×4 blocks [4]. Then we reduced the dimension of the feature vectors down to 128 by PCA.

We evaluated classification performances when filtering the prototype dataset, i.e. retaining a proportion θ of the most relevant examples as prototypes for classification. Such a data reduction capability is one of the most interesting properties of UNN, as it favourably impacts on the computational cost of classification, which grows at least logarithmically (at most linearly) with the dataset size. Indeed, classification roughly amounts to searching for the k nearest neighbors among prototypes, which is $O(kd\theta m)$ for linear exhaustive search, $O(kd\log(\theta m))$ (at best) for fast kD-tree based search [AMN+98].

In Fig. 4.12 and 4.13 we show classification performances in terms of the mean Average Precision (mAP) [5] as a function of θ. 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. Furthermore, we also removed prototypes with

[4] We used the matlab implementation by the author, which is publicly available at http://people.csail.mit.edu/torralba/code/spatialenvelope/sceneRecognition.m

[5] 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.

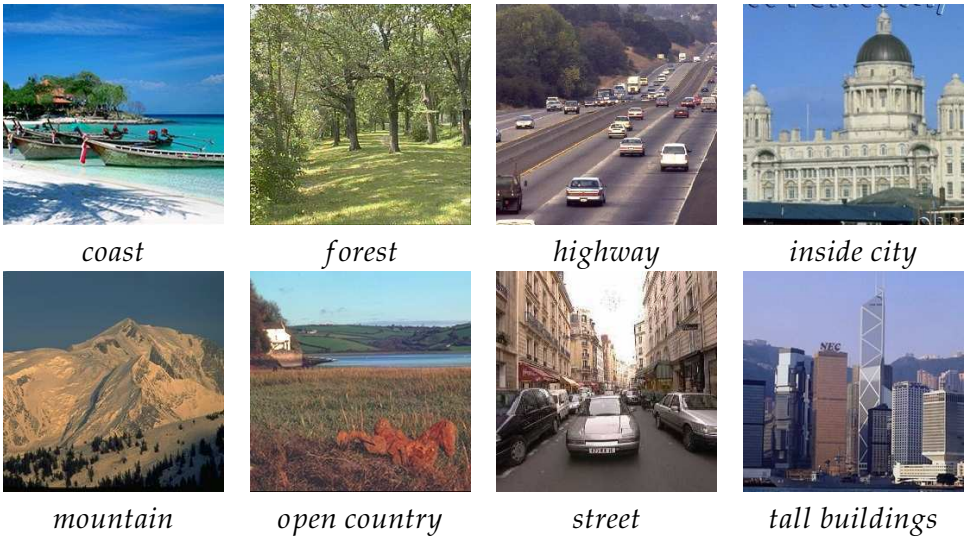


Figure 4.10: *Examples of annotated images from the 8 categories database of [OT01].*

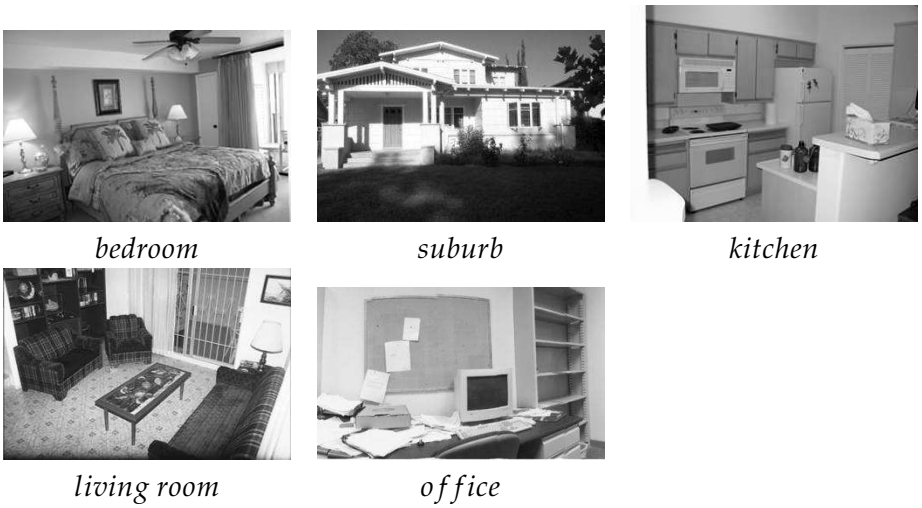


Figure 4.11: *Examples of the five additional categories included in the 13 categories database of [FFP05].*

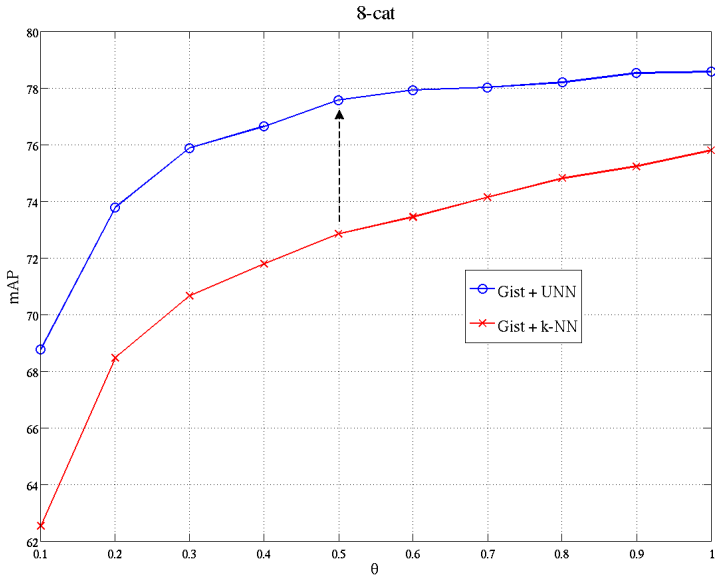


Figure 4.12: *Gist image classification performances of* UNN *compared to k-NN on the 8-cat database.*

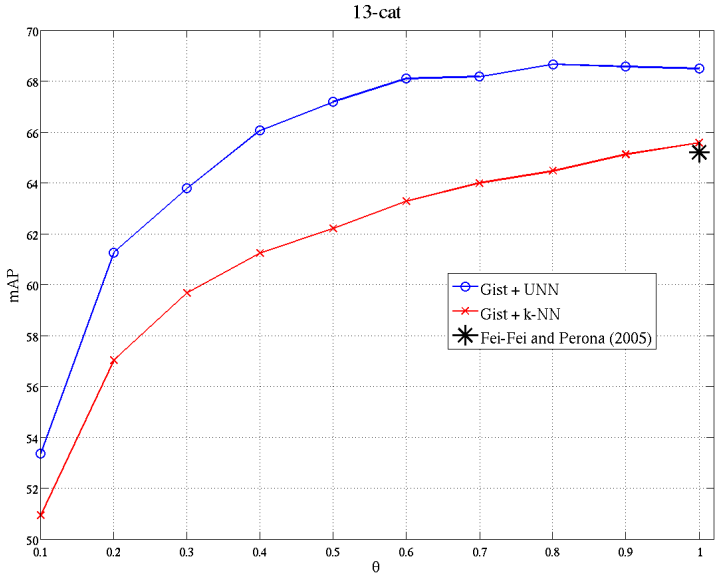


Figure 4.13: *Gist image classification performances of* UNN *compared to k-NN on the 13-cat database.*

negative leveraging coefficients, thus constraining all the retained classifiers to have $\alpha_{j\bar{c}} > 0$ (where $\bar{c}$ is the true class of prototype j).

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 θ (between 10% and the whole set of examples). UNN significantly outperforms classic k-NN and the trend observed on the synthetic data when filtering the prototype set is confirmed on image categorization results. Indeed, selecting a reduced set of prototypes limits over-fitting on training data, while improving classification performance on the test set up to 5%. Namely, consider performances at $\theta = 0.5$ in Fig. 4.12, where UNN yields the most interesting result, outperforming the best k-NN performance by almost 2% with 50% of data. Moreover, on the 13-cat database our method outperforms the technique proposed by [FFP05] by 3% (look at the asterisk in Fig. 4.13, which corresponds to the best result reported in their paper).

To summarize, UNN displays the ability to discriminate the most relevant images of each class, thus inducing a classification rule robust to "noisy" prototypes arising from low inter-class variations. Adjusting the value of θ enables to remove those confusing prototypes, thus reducing the representation of each category to a sparse subset of meaningful prototype images.

Fig. 4.14 shows two examples of how the leveraged k-NN rule may correct misclassifications due to the uniform k-NN voting. *E*.g., in the first example, the classic and the boosted k-NN methods are compared when classifying an image belonging to class *coast*, with $k = 11$. The leveraged rule with as few as 20% of prototype images is able to correctly label the query image (first row). Below each nearest neighbor image we show its contribution to the classifier of (4.10): note that negative votes are significantly smaller than positive ones (up to an order of magnitude), thus determining positive labeling with high prediction score h_c^ℓ, according to (4.10). On the contrary, uniform voting rule with all prototypes misclassifies the test image, not being able to reject contributions by "noisy" neighbor images. An example of prototypes selected by filtering the dataset is shown in Fig. 5.9, where the leveraging coefficients refer to the first category (*tall buildings*) versus the remaining ones.

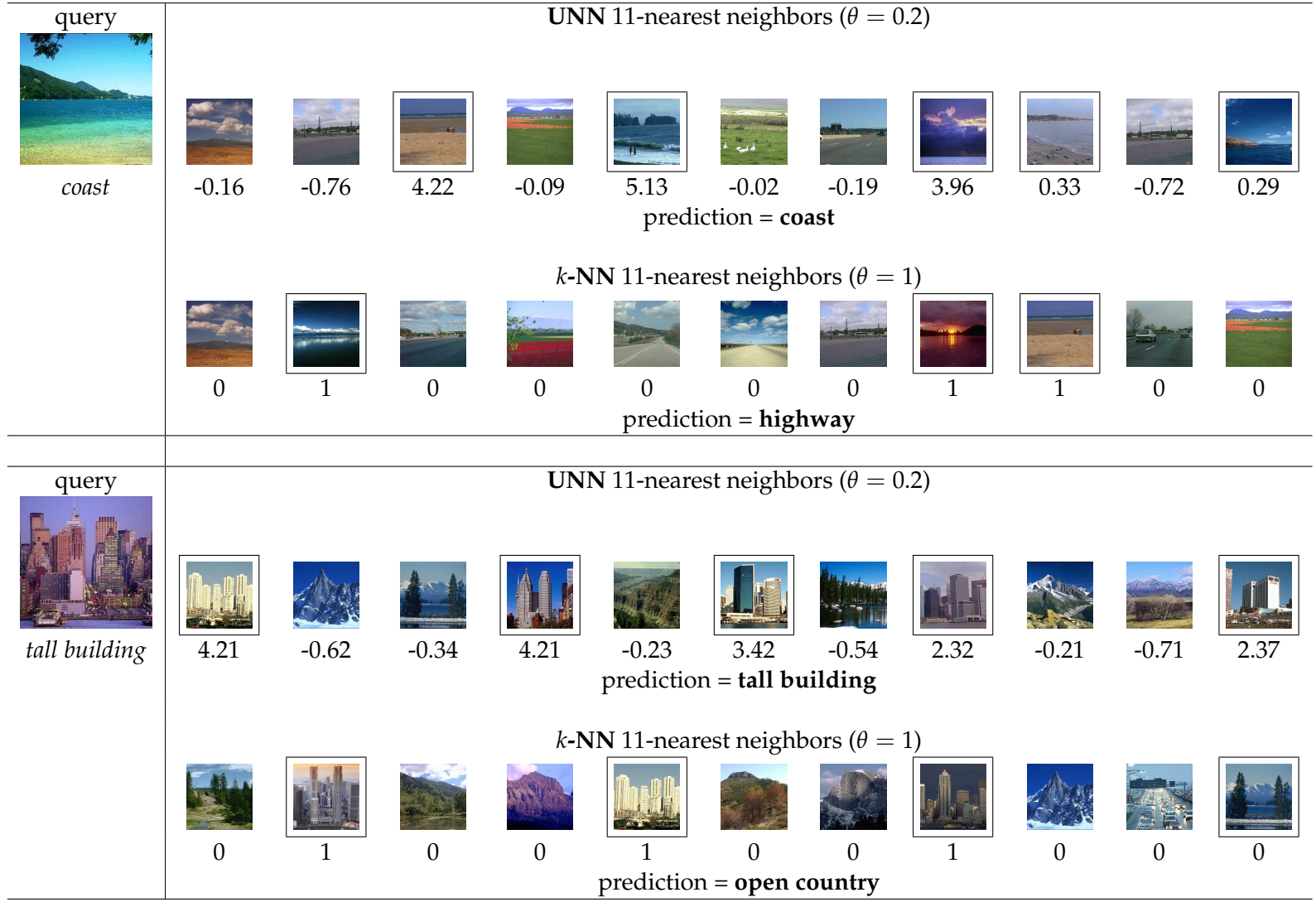


Figure 4.14: *Two examples where* UNN *corrects misclassifications of k-NN. The query image is shown in the leftmost column. The 11-nearest prototype images are shown on the right: the first row refers to* UNN *with 20% of retained prototypes ($\theta = 0.2$), whereas the second column refers to classic k-NN classification over all prototypes ($\theta = 1$).*

4.3.4 Image categorization based on Bag-of-Features

We validated the ability of UNN to "boost" 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" [SZ03]. An image is then represented by the histogram of visual word frequencies. This approach provides an effective tool for image categorization, as it relies on one single compact descriptor per image, while keeping the informative power of local features.

In this section we describe categorization based on Bag-of-Features and compare UNN with both classic k-NN voting and a state-of-the-art classification method on the 8-cat database (Sec. 4.3.3). We used the experimental setup of [BZM08] and reproduced their method using the probabilistic Latent Semantic Analysis (pLSA) implementation of [SRE+05] [6]. So as for the local descriptors, we used the VLFeat toolbox [VF08] [7] 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 256×256, 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 1,500 visual words, which was found to be optimal by [BZM08].

In Fig. 4.15 we show performances in terms of mAP as a function of the proportion θ of retained prototypes (like in Sec. 4.3.3). Namely, results of UNN on BoF histograms are compared to those of classic k-NN on the same descriptors. Futhermore, pLSA descriptors were learned from these BoF histograms and used for k-NN classification as well (we set $Z = 25$ pLSA topics following [BZM08]). UNN outperforms the two compared methods by a significant amount, improving precision by 5% to 10% over k-NN. The gain over pLSA is also significant (3% to 5%). In particular, notice that UNN can reach performances equal or superior to those of k-NN for very small values of θ, thus significantly decreasing the computation time for classification. For instance, look in Fig. 4.15(a) at the point on the blue

[6] Code available at http://www.robots.ox.ac.uk/~vgg/software/pLSA/pLSA_demo.tgz.

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

curve for $\theta = 0.3$, where UNN outperforms the best k-NN accuracy by 3%, while matching the best pLSA result despite using only 30% of the data. The same phenomenon is observed also in Fig. 4.15(b), where we used L2-normalized Bag-of-Features histograms, instead of performing the standard L1 normalization as in Fig. 4.15(a). In this second case the gap between UNN and the two other methods is at most 5%, but our approach still performs the best and, most importantly, it is much more robust to histogram normalization. (Notice that the UNN curve in Fig. 4.15(b) looks very similar to that in Fig. 4.15(a).)

To summarize, in spite of its simplicity and the reduced computational cost due to selecting sparse prototypes, UNN is able not only to significantly "boost" k-NN classification, but even to outperform a far more computationally expensive and sophisticated method like pLSA, which relies on fitting a probabilistic model in an unsupervised way.

Finally, the improvement of UNN over classic uniform voting is significant even when using a different metric for the nearest neighbor matching. In Fig. 4.15(c) we tested the Histogram Intersection matching criterion. This similarity measure was firstly proposed by [SB91] for image indexing based on color histograms, and, more recently, it has been successfully used by [LSP06] in the context of Bag-of-Features image categorization. The Histogram Intersection criterion turns out to be more appropriate than the Euclidean distance when measuring similarities between L1-normalized Bag-of-Features descriptors. In particular, notice that results of either k-NN or UNN with this setting improve everywhere those reported in Fig. 4.15(a) and 4.15(b) for the Euclidean setting (up to 5% gap for k-NN and 7% for UNN). One more time, the most interesting point on the UNN curve refers to selecting a sparse prototype subset: *e.g.*, retaining only 20% of prototypes ($\theta = 0.2$) gives roughly the same precision as k-NN with the entire training set. A general comparison between UNN and k-NN with the three different settings is depicted in Fig. 4.16. Notice that our method using the Histogram Intersection matching outperforms all the compared curves for as few as 30% of prototypes. *E.g.*, consider the point on the upper black curve for $\theta = 0.3$, which corresponds to the best precision-computation time trade-off.

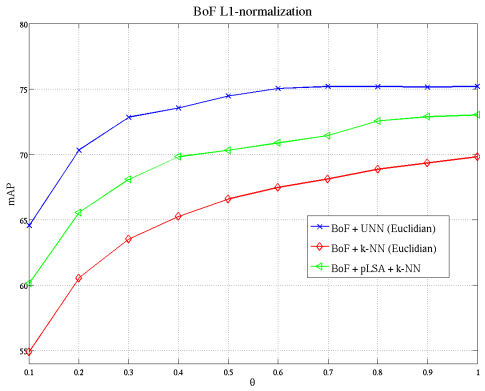
§ 4.4 Conclusion

In this chapter, we contribute to fill an important void of k-NN methods, showing how boosting can be transferred to k-NN classification. Namely, we have proposed a novel boosting algorithm, UNN (Universal Nearest Neighbors rule), for inducing a leveraged k-NN rule. This rule generalizes classic k-NN to weighted voting where weights, the so-called leveraging coefficients, are iteratively learned by UNN. We prove that this algorithm

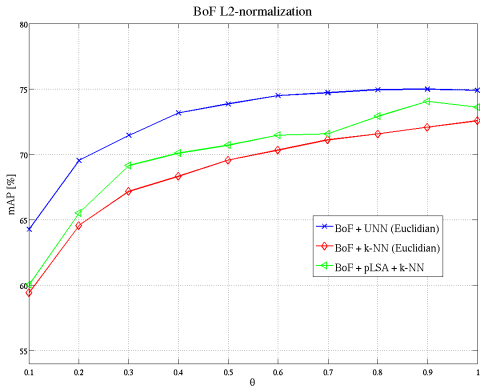
converges to the global optimum of surrogate risks under very mild assumptions.

Experiments on both synthetic and image categorization databases display that UNN provides significant performance improvements (up to the best possible performance of the Bayes rule), as well as consistent data reduction ability, which results in significant speed-ups for classification (up to a factor 2 when removing half of the coefficients).

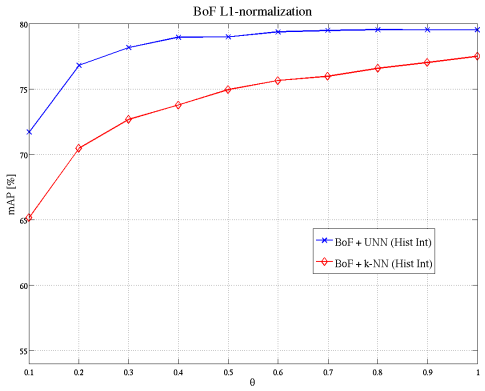
Our approach is built on the top of k-NN search, thus being fully compatible with most existing techniques relying on metric distance learning [ZBMM06] as well as PCA or kernel transformations of the input space, which are expected to enable significant improvements of categorization performances [Jai10].



(a)



(b)



(c)

Figure 4.15: *Performances of image categorization with Bag-of-Features (BoF) on the 8-cat database: the mean Average Precision (mAP) is shown as a function of the proportion θ of instances that are retained for classification. (a) L1 normalization of BoF histograms and k-NN search wrt Euclidean distance. (b) L2 normalization of BoF and k-NN search wrt Euclidean distance. (c) L1 normalization of BoF and k-NN search wrt Histogram Intersection kernel.*

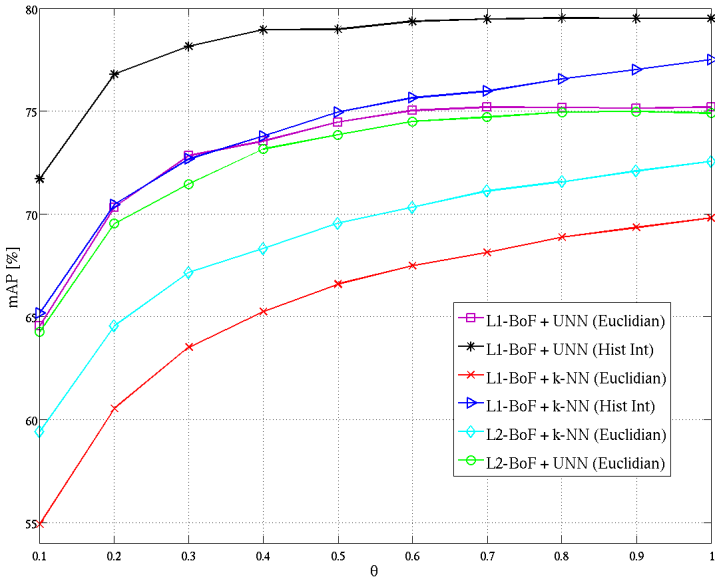


Figure 4.16: *Overall results of BoF classification with* UNN *compared to k-NN for different settings of histogram normalization (either L1 or L2-norm) and nearest neighbor matching (either Euclidean distance or Histogram Intersection).*

PART III

LEVERAGING MULTI-CLASS KERNEL DENSITY CLASSIFICATION

- CHAPTER 5 -

MLNN: MULTI-CLASS LEVERAGED k-NN

In the previous chapter, we have introduced the task of real-world image categorization in a *single-label* supervised learning framework. In particular, we have focused on *prototype-based* methods like the k-NN voting rule, which provides an effective tool for classifying images when relying on appropriate descriptors of the visual information, In order to enforce supervision in such methods so as to constrain the test error under certain reasonable bounds, we have generalized the classic voting rule to a *leveraged* k-NN voting rule. In this framework, only the most relevant annotated instances are taken into account for the classification of test data, as their confidence is learned during the training phase by our UNN boosting-like algorithm. Therefore, UNN allows us both to "boost" the classification accuracy by weighting contributions of different examples according to their prototypical relevance (*importance weighting*), and improve the computational efficiency thanks to the sparsity of the prototype set selected for testing.

However, two major issues still remain unsolved in our UNN approach. The first is related to the binary formulation of the learning problem, *i.e.*, training classifiers on *positive* against *negative* examples, as inspired by the classic boosting formulation (*e.g.*, AdaBoost). In this framework, the multi-class categorization task has to be redefined as multiple *binary* classification problems. *E.g.*, in Chap. 4, we have presented results of image categorization using UNN in a *one-versus-all* framework, where examples of a fixed category ("positives") are trained against all the remaining examples ("negatives"). This formulation significantly impacts on computational time of both training and classification phase, which is linear in the number of categories for a fixed dataset size.

The second issue is related to the classification performances, which are particularly sensitive to the sparsity of the prototype set, thus possibly affecting the k-NN prediction accuracy when data are too sparse in a high

dimensional feature space. This problem can be viewed from a statistical point of view, as the issue of improving the local class density estimation that underlies the k-NN voting rule.

In order to tackle these two problems, in this chapter we consider a *multiclass, kernel-based* generalization of UNN, which we call MLNN (Multiclass Leveraged k-Nearest Neighbors). Indeed, the first issue is tackled by modifying the risk function so as to take into account a single *edge* defined over all classes simultaneously, thus exploiting the intrinsic multiclass nature of k-NN classifiers (this is a major advantage of using k-NN over more sophisticated approaches, which are instead constrained to binary classification, like SVM). Furthermore, we address the second issue by proposing a new formulation of the instance-based classification that grounds on a statistical interpretation of k-NN as *maximum-a-posteriori* classification relying on non-parametric kernel-based density estimation. This consideration allows one to generalize from the case of the k-NN kernel, which has the major drawback of being too much local (thus too sensitive to setting of k), as well as that of being uniform (thus not taking into account the geometric configuration of neighborhoods in the feature space), to generic kernels tailored for density estimation, like the Radial Basis Function (RBF) kernels, which is commonly used in SVM as well. According to this interpretation we generalize our boosted k-NN classifier to boosting generic kernels for class density estimation. Furthermore, such a general formulation of our learning problem closely resembles that of SVM, as it consists in optimizing a surrogate risk defined over training data. The main differences are the nature of the cost functional to minimize, the minimization procedure that, being inspired by the boosting theory, is more straightforward, and the regularization term, which in our case is implicitely related to the number of boosting iterations, thus allowing directly to set the sparsity degree of the solution, which is in general not guaranteed to be significant enough in the case of classic SVM. In this chapter we show that the main theoretical results shown in Chap. 4 extend naturally to the multiclass case, as well as to using kernels instead of k-NN. Therefore, we prove that, under mild assumptions, MLNN converges fast to the *global optimum* of our multiclass exponential surrogate risk functional.

We tested MLNN on three datasets of natural images. Results obtained using Gist descriptors show that MLNN significantly outperforms classic k-NN and weighted k-NN voting. Furthermore, using an adaptive kernel with MLNN provides significant accuracy improvement on such images. Moreover, our multiclass learning technique dramatically reduces the computation time by a factor C (number of classes), compared to the classic one-versus-all approach. MLNN accuracy is also comparable with state of the art metric learning method. Since MLNN is fully compatible with these metric learning method, the best accuracy is obtained when using MLNN with the optimal metric. Most interestingly, MLNN is able not only to re-

duce the computational cost with respect to one-vs-all classification, but also to improve the accuracy, and this gain becomes as more significant as the number of classes gets larger.

Then, we carried out experiments using the state-of-the-art Bag-of-Features descriptors, showing that choosing the right normalization and similarity kernel may significantly improve classification performances. We quantify the advantage of multi-class MLNN classification over the binary UNN in terms of both precision and computational cost. Furthermore, we compare our results to SVM (both linear and kernel-based), showing that MLNN performances, although being inferior, are obtained at a significantly lower computational cost, thus possibly providing a more reasonable precision/cost trade-off when managing huge image collections.

§ 5.1 INTRODUCTION

5.1.1 Inherently multi-class learning

Recent research has devoted much effort to tailoring most existing learning algorithms to *multi-class* classification problems in such a way that classification boundaries between all the classes could be learned at the same time. Such classification approaches are *inherently* multi-class and they do not require splitting the multi-class problem into multiple binary problems, thus favourably impacting on the computational time when dealing with many categories. Nevertheless, redefining discriminative methods in an inherently multi-class framework is challenging, as their classic formulation generally allows only to fit a boundary between two classes of positive vs negative examples. Indeed, for many data mining and computer vision applications, the standard approach to multi-class problems is to split them into several independent two-class problems. Generally, this is accomplished by either training classifiers on two distinct classes at a time (*one-versus-one* methods), thus learning a boundary between each pair of classes, or redefining the example labels in such a way that one class is considered as positive, whereas all the others are grouped into a single "negative" class (*one-versus-all* methods). This latter kind of approaches is more convenient from a computational point of view, as the cost of training is linear in the number of categories, while one-versus-one training is quadratic. Finally, a more general class of methods for splitting multi-class learning problems into binary problems in based on Error-Correcting Output Codes (ECOC), which have been extensively described by Allwein et al [ASS00].

Besides strategies relying on problem splitting, several methods have been also proposed, which try to make binary algorithms able to deal with many classes simultaneously. Namely, in the context of boosting literature, the simplest approach is AdaBoost.M1 [FS96]. This is a straightforward ex-

tension of the classic AdaBoost to multi-class learning, where weak classifiers are allowed to give multi-class predictions. The limit of this approach is that the same constraint on the accuracy of weak classifiers is imposed as in classic AdaBoost, *i.e.*, each classifier is required to have an accuracy larger than 50%, which is as more difficult to be matched as the number of classes gets larger. In order to relax this hard constraint, a very effective approach has been recently proposed by Zhu et al [ZRZH09] for generalizing AdaBoost to multi-class classifiers. This method is called SAMME (Stagewise Additive Modeling using a Multiclass Exponential loss), and is grounded on a statistical interpretation of AdaBoost as an iterative strategy for fitting a linear expansion of the classification function using elementary functions ("weak classifiers"). The main advantage of SAMME is to require that each weak classifier be only slightly more accurate than random guessing, thus making it easier to fit the classification function using "off-the-shelf" multi-class classifiers. The trick behind this strategy is to define an inherently multi-class loss function that operates on a suitable classification margin, so as to "balance" loss contributions from positive and negative losses (they call this condition "symmetry" referred to the target variables). The formulation of SAMME algorithm looks very similar to that of classic AdaBoost, thus representing its most natural extension for dealing with multi-class classifiers.

Inspired by the SAMME approach of Zhu et al [ZRZH09], we provide a multi-class generalization of our UNN approach (described in Chap. 4). In particular, our method, called Multiclass Leveraged k-NN (MLNN), consists in a framework for inducing a generalized multi-class k-NN classifier by minimizing an inherently multi-class risk function over training data. The key aspect when generalizing from binary to multi-class learning is properly defining a *multi-class edge* as a primer for the minimization of the training error. Indeed, this edge takes into account the accuracy of induced classifiers over all classes simultaneously (Sec. 5.2.3).

5.1.2 k-NN class density estimator

The k-NN rule is one of the simplest classification methods, though it turns out to be still very attractive to practitioners in many application domains, among which computer vision algorithms. Interestingly, besides the intuitive meaning of using the closest prototypes for predicting the unknown labels of observations, a well-founded statistical intepretation of k-NN can be formulated. Indeed, as pointed out by Duda and Hart [DHS01], k-NN classification can be viewed as a particular case of maximum-a-posteriori classification, where one obtaines estimates of the local class density *directly* from the annotated data, using the k-NN window for local non-parametric density estimation. Roughly speaking, using the k-NN rule for classifying unseen data is equivalent to counting the proportion of the k-nearest neigh-

bor prototypes that belong to each class, *i.e.*, the class density estimates around those data, and then taking the labels maximizing these scores as the final class predictions.

However, k-NN as a density estimator is part of a broader class of non-parametric methods that have been deeply investigated in statistics. In particular, the most interesting class of such estimators is represented by multivariate *kernel density estimators*, *i.e.*, estimation functions that rely on similarity kernels in order to provide pointwise estimates of probability densities. Following the general tractation of Terrell and Scott [TS92], we can define a generic kernel estimator as the following function:

$$\hat{f}(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h(\cdot)^d} K\left(\boldsymbol{x}, \boldsymbol{x}_i; h(\cdot)\right) \ , \tag{5.1}$$

where $\boldsymbol{x}$ is the estimation point, $\boldsymbol{x}_i$ are the samples and n their number, whereas h is a parameter related to the kernel scale, whose argument may vary. In particular, dependingly on the argument of this parameter, three main kinds of estimation kernels can be defined:

- *Parzen window*, when h is a constant parameter, *i.e.*, independent from the data. This is equivalent to considering a *fixed-size* window centered at $\boldsymbol{x}$ that weights samples' contributions.
- *Balloon estimator*, when $h = h(\boldsymbol{x})$, *i.e.*, the kernel scale is *locally adapted* around the estimation point. This amounts to considering a window centered at $\boldsymbol{x}$ whose scale depends – say – on the local density of samples.
- *Sample-point estimator*, when $h = h(\boldsymbol{x}_i)$, *i.e.*, the kernel scale varies for *each* sample. This is equivalent to computing a mixture of identical but individually scaled kernels, each one being centered at a different observation $\boldsymbol{x}_i$.

k-NN can be viewed as a special case of the balloon estimator, with uniform value inside a sphere centered at the estimation point, whose radius equals the k-NN distance. Thus, k-NN kernel writes as follows:

$$\tilde{K}_{\mathrm{NN}_k}(\boldsymbol{x}, \boldsymbol{x}_i) = \frac{d}{S_d} \left[\frac{||\boldsymbol{x} - \boldsymbol{x}_i||}{\rho_k(\boldsymbol{x})} \leq 1 \right] \ , \tag{5.2}$$

where d is the dimension of the feature space, S_d the hyper-surface area of a d-sphere of unit radius, ρ_k is the k-NN distance to $\boldsymbol{x}$ and square brackets denote the indicator function. (Term $\frac{d}{S_d}$ is only a normalization factor, which is necessary for obtaining a true probability estimate.) Notice that in (5.2) ρ_k plays the role of a scale parameter that, for fixed sample dataset, depends only on the estimation point. Such a "hard" windowing function

can be also used for the sample-point estimator, thus giving rise to the so-called *reciprocal* k-NN estimator, where the only not-null contributions are those from samples that are the reciprocal k-nearest neighbors of the estimation point. In this case, the kernel definition looks very similar to (5.2), but the scale parameter varies for every sample point:

$$\tilde{K}_{\mathrm{RNN}_k}(\boldsymbol{x}, \boldsymbol{x}_i) = \frac{d}{S_d} \left[\frac{||\boldsymbol{x} - \boldsymbol{x}_i||}{\rho_k(\boldsymbol{x}_i)} \leq 1 \right] . \tag{5.3}$$

Also a mixed solution between the two has been proposed by Nock et al [NSJ00], called *symmetric* k-nearest neighbors, which are obtained by combining contributions from both k-NN of (5.2) and reciprocal k-NN of (5.3) (logic sum of the two conditions in square brackets). However, notice that computing (5.3) is much more computationally expensive than (5.2), because evaluating this kernel at a single estimation point $\boldsymbol{x}$ requires to search for the k-nearest neighbors of *each* sample. This is why non-symmetric k-NN is preferable in many cases, although its behaviour may be less robust [NSJ00].

Independently on which particular kernel is used, estimator (5.1) can be straightforwardly applied to *maximum-a-posteriori* classification, by computing it separately on each class of annotated samples. Indeed, denoting with $c_i = \{1, 2, \ldots, C\}$ the ground-truth annotation of sample $\boldsymbol{x}_i$ and m the total number of annotated data available, we can estimate the joint probability density of observations and classes as follows:

$$p_m(\boldsymbol{x}, c) = \frac{1}{m} \sum_{i:c_i=c} \frac{1}{h(\cdot)^d} K\left(\boldsymbol{x}, \boldsymbol{x}_i; h(\cdot)\right) . \tag{5.4}$$

Thus, using the Bayes' rule, we have the following estimation of the a-posteriori probability of class c given $\boldsymbol{x}$:

$$P_m(c|\boldsymbol{x}) = \frac{p_m(\boldsymbol{x}, c)}{\sum_{j=1}^{C} p_m(\boldsymbol{x}, j)} = \frac{\sum_{i:c_i=c} \frac{1}{h(\cdot)^d} K\left(\boldsymbol{x}, \boldsymbol{x}_i; h(\cdot)\right)}{\sum_{i=1}^{m} \frac{1}{h(\cdot)^d} K\left(\boldsymbol{x}, \boldsymbol{x}_i; h(\cdot)\right)} . \tag{5.5}$$

This is the general formulation of the class probability estimation when using kernels, which can be directly used for classification purposes using the maximum-a-posteriori rule. In the special case of k-NN classification, *i.e.*, plugging kernel (5.2) into (5.5), we find the common definition:

$$P_m^{k\mathrm{NN}}(c|\boldsymbol{x}) = \frac{\sum_{i:c_i=c} K\left(\boldsymbol{x}, \boldsymbol{x}_i; h(\cdot)\right)}{\sum_{i=1}^{m} K\left(\boldsymbol{x}, \boldsymbol{x}_i; h(\cdot)\right)} = \frac{k_c}{k} , \tag{5.6}$$

where k_c is the number of k-NN of $\boldsymbol{x}$ that belong to class c.

5.1.3 Supervised kernel density learning

In the following section, we use the generic formulation (5.5) of maximum-a-posteriori classification using density estimation kernels as the basis for our MLNN method. In particular, our method addresses the problem of learning the "importance" of sample points for improving the robustness of MAP classification using (5.5). Therefore, we propose a boosting strategy for leveraging the contributions to the probability estimation from different prototypes, while removing those instances that are noisy, thus less reliable. The main idea behind our approach is to "fuse" the non-parametric class density estimation problem with the boosting approach, thus exploiting the benefits of discriminative learning as provided by boosting in terms of generalization power and robustness to noisy data.

The problem of embedding discriminative approaches into a generative framework has been tackled by recent literature in different ways, *e.g.*, by representing support vector machines in the framework of Gaussian mixture densities, as proposed by Deselaers et al [DHN10]. Namely, they present an approach to fuse SVM with a Gaussian mixture density classifier trained in a generative framework, thus adding robustness to the generative model while reducing the risk of overfitting that may affect the discriminative part. Also boosting algorithms have been demonstrated very useful to improve generative models for both density estimation (unsupervised learning) and classification (supervised). On the one hand, Rosset and Segal [RS02] have applied the boosting approach to the unsupervised learning problem of density estimation, particularly focusing on using Bayesian networks as weak learners. On the other hand, several methods have been proposed that use generative models like mixtures of Gaussians as base learners for boosting density-based classification [LWZZ06]. Our approach follows the aim of this research trend, exploiting density kernel classifiers in a *multi-class, leveraged* way. In particular, prototypes are interpreted as sample points for a kernel density estimation procedure, whereas a boosting algorithm learns the best linear combination of them in a discriminative way, *i.e.*, minimizing an inherently multi-class risk function. Thus, this strategy provides a multi-class classification rule which "fits" a non-parametric sample density using sparse prototypes of the different classes. In this framework of boosted prototype-based classification, the UNN approach presented in Chap. 4 can be easily viewed as a particular case that arises from boosting a particular kernel density classifier (5.6) for binary learning.

In this chapter, we test our MLNN approach on the challenging task of real-world image categorization, showing its ability to significantly improve vanilla k-NN on this task. MLNN does not need to learn a distance function, as it directly operates on the top of k-nearest neighbors search. At the same time, it does not require an explicit computation of the fea-

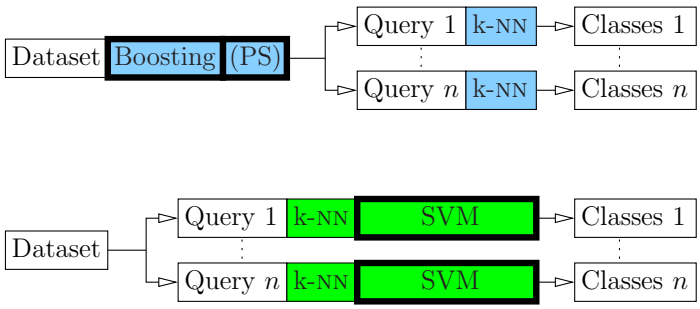


Figure 5.1: *Optimizing k-NN via* MLNN *(up, blue) and SVM-KNN [ZBMM06] (down, green).* MLNN *uses a boosting algorithm before being presented any query, while SVM-KNN learns support vectors after each query is presented. Bold rectangles indicate induction steps (PS = prototype selection; see text for details).*

ture space, thus preserving one of the main advantages of prototype-based methods. Compared to other local learning approaches to k-NN classification, like that of Zhang et al [ZBMM06], MLNN also speeds up query processing: instead of learning a local classifier for each query, MLNN performs learning upwards, once and for all, and does not need to be run again or updated depending on queries (Figure 5.1). Finally, the most significant advantage of MLNN lies in its ability to find out the most relevant prototypes for categorization, allowing to filter out the other examples. Experimentally, significant data reductions are observed with a simultaneous increase in categorization performances.

In the following section we describe MLNN, along with the statement of its theoretical properties. This section is followed by an experimental section displaying the behavior of MLNN on a standard image categorization database using the two best known state-of-the-art image descriptors (*i.e.*, Gist and Bag-of-Features). A last section concludes. Finally, in Appendix B, we provide the general formulation of our algorithm for the minimization of any multi-class surrogate risk matching the same conditions as for UNN. We discuss the solution of the learning equation more formally and we give the specialization of our algorithm for the case of logistic loss.

§ 5.2 METHOD

5.2.1 Problem statement and notations

In this chapter, we address the image classification task in an inherently multi-class way. Thus, instead of splitting the multi-class problem in as many *one-versus-all* (two-class) classification problems – which is a frequent

approach in boosting [SS99] – we tackle directly the *multi-class* problem, following Zhu et al [ZRZH09]. Indeed, for a given query image, we compute its *classification score* for all categories. While we basically use this vector for single-label prediction using the category with the maximum score, our algorithm can be straightforwardly extended to multi-label prediction and ranking [SS99]. We suppose given a set $\mathcal{S}$ of m annotated images. Each image is a training *example* $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ is the image feature vector and $\boldsymbol{y}$ the *class vector* that specifies the category membership of the image. In particular, the sign of component y_c gives the positive/negative membership of the example to class c ($c = 1, 2, ..., C$), 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 Zhu et al [ZRZH09], we constrain the class vector to be *symmetric*, *i.e.*:

$$\sum_{c=1}^{C} y_c = 0 \ , \tag{5.7}$$

by setting: $y_{\tilde{c}} = 1$, $y_{c \neq \tilde{c}} = -\frac{1}{C-1}$, where $\tilde{c}$ is the true image category. Furthermore, we denote by $K(\boldsymbol{x_i}, \boldsymbol{x_j})$ a symmetric similarity kernel between two examples $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$.

5.2.2 (Leveraged) kernel density classification

The vanilla k-NN rule is based on majority vote among the k-nearest neighbors in set $\mathcal{S}$, in order to predict the class of query $\boldsymbol{x}_q$. It can be defined as the following multiclass classifier $\boldsymbol{h} = \{h_c,\ c = 1, 2, ..., C\}$:

$$h_c(\boldsymbol{x}_q) = \frac{1}{k} \sum_{i\,:\,\boldsymbol{x}_i \in \mathrm{NN}_k(\boldsymbol{x}_q)} [y_{ic} > 0] \ , \tag{5.8}$$

where $h_c \in [0, 1]$ is the classification score for class c, $\mathrm{NN}_k(\boldsymbol{x})$ denotes the set of k-nearest neighbors of observation $\boldsymbol{x}$ and square brackets denote the indicator function.

Eq. (5.8) is equivalent to the following:

$$h_c(\boldsymbol{x}_q) = \frac{1}{k} \sum_{i=1}^{m} \left[\frac{||\boldsymbol{x}_q - \boldsymbol{x}_i||}{\rho_k} \leq 1 \right] [y_{ic} > 0] \ , \tag{5.9}$$

where the first term inside the summation equals kernel (5.2), up to a constant normalizing factor. In the following, for the sake of simplicity, we only consider *unnormalized* kernels, like the following:

$$K_{\mathrm{NN}_k}(\boldsymbol{x}, \boldsymbol{x}_i) = \left[\frac{||\boldsymbol{x} - \boldsymbol{x}_i||}{\rho_k(\boldsymbol{x})} \leq 1 \right] \ . \tag{5.10}$$

For the purpose of density-based classification, it is straightforward to verify that the "positive" prediction rule (5.8) can be equivalently transformed into the following "real" classification rule, thus not affecting the final predictions:

$$h_c^{\mathrm{NN}_k}(\boldsymbol{x}_q) = \sum_{i=1}^{m} \frac{1}{k} K_{\mathrm{NN}_k}(\boldsymbol{x}_q, \boldsymbol{x}_i) y_{ic} \ , \tag{5.11}$$

where $h_c^{\mathrm{NN}_k} \in \left[-\frac{1}{C-1}, 1\right]$, due to the definition of class vectors (Sec 5.2.1).

In this chapter, we propose to generalize rule (5.11) to the following *leveraged kernel density* classification rule $\boldsymbol{h}^\ell = \{h_c^\ell\}$:

$$h_c^\ell(\boldsymbol{x}_q) = \sum_{j=1}^{T} \alpha_j K(\boldsymbol{x}_q, \boldsymbol{x}_j) y_{jc} \in \mathbb{R} \ , \tag{5.12}$$

where prediction h_c^ℓ takes values in $\mathbb{R}$. In (5.12), we have introduced the three following elements to generalize (5.11):

- *leveraging coefficients* α_j, that provide a *weighted* voting rule instead of uniform voting;
- generic kernel K, which takes into account "soft" (real-valued) similarities between query $\boldsymbol{x}_q$ and prototypes $\boldsymbol{x}_j$, instead of "hard" selection of the most similar (k-NN) prototypes;
- size T of the set of *prototypes* that are allowed to vote.

This last point is particularly interesting for computational purposes, as our classification rule actually involves only a (possibly sparse) subset of the training data as prototypes to be used at query time. Indeed, a *prototype selection* step is to be performed while training our classifier, in order to determine the most relevant subset of training data, *i.e.*, the so-called *prototypes*, forming a set $\mathcal{P} \subseteq \mathcal{S}$ (Figure 5.1). The prototypes are selected during the training phase, which consists in fitting their coefficients α_j, while removing the least relevant annotated data from $\mathcal{S}$.

5.2.3 Multi-class surrogate risk minimization

In order to fit our classification rule (5.12) onto training set $\mathcal{S}$, we focus on the minimization of a multi-class exponential (surrogate[1]) risk:

$$\varepsilon^{\exp}\left(\boldsymbol{h}^\ell, \mathcal{S}\right) \doteq \frac{1}{m} \sum_{i=1}^{m} \exp\left\{-\rho(\boldsymbol{h}^\ell, i)\right\} \ , \tag{5.13}$$

[1] We call *surrogate* a function that upperbounds the risk functional we should minimize, and thus can be used as a primer for its minimization.

where:

$$\rho(\boldsymbol{h}^{\ell}, i) = \frac{1}{C}\sum_{c=1}^{C} y_{ic} h_c^{\ell}(\boldsymbol{x_i}) \tag{5.14}$$

is the multiclass *edge* of classifier $\boldsymbol{h}^{\ell}$ on training example $\boldsymbol{x_i}$. This function is an upper bound of the *empirical risk*:

$$\varepsilon^{0/1}\left(\boldsymbol{h}^{\ell}, \mathcal{S}\right) \doteq \frac{1}{m}\sum_{i=1}^{m}\left[\rho(\boldsymbol{h}^{\ell}, i) < 0\right] \ . \tag{5.15}$$

Minimizing (5.13) acts as a convenient primer for the minimization of (5.15) — convenient as (5.15) is not differentiable and often computationally hard to minimize [NN09b, NN09c]. Both risks (5.13, 5.15) depend on quantity $y_{ic}h_c^{\ell}(\boldsymbol{x_i})$, that is the *edge* of classifier $\boldsymbol{h}^{\ell}$ on example $(\boldsymbol{x_i}, \boldsymbol{y_i})$ for class c. This edge is positive iff the category membership predicted by the classifier agrees with the true membership of the example. The surrogate risk (5.13) uses a *multi-class edge*, which averages all the edges for the C classes. In order to minimize (5.13), we propose a boosting-like procedure, *i.e.*, an iterative strategy where the strong classifier (5.12) is updated as follows:

$$h_c^{(t)}(\boldsymbol{x}_i) = h_c^{(t-1)}(\boldsymbol{x}_i) + \delta_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) y_{jc} \ , \tag{5.16}$$

with j being the index of weak classifier chosen at iteration t. Plugging (5.16) into (5.14), and then into (5.13), turns the optimization problem to finding δ_j that minimizes the following objective:

$$\arg\min_{\delta_j} \sum_{i=1}^{m} w_i \cdot \exp\left\{-\delta_j \mathrm{r}_{ij}\right\} \ , \tag{5.17}$$

where w_i is a weighting factor depending on past weak classifiers:

$$w_i = \exp\left\{-\frac{1}{C}\sum_{c=1}^{C} y_{ic} h_c^{(t-1)}(\boldsymbol{x_i})\right\} \ , \tag{5.18}$$

and r_{ij} is a pairwise term only depending on data and kernel:

$$\mathrm{r}_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)\frac{1}{C}\sum_{c=1}^{C} y_{ic} y_{jc} \ . \tag{5.19}$$

Finally, taking the derivative of (5.17), the global minimization of surrogate risk (5.13) amounts to fitting δ_j so as to solve the following equation:

$$\sum_{i=1}^{m} w_i \mathrm{r}_{ij} \exp\left\{-\delta_j \mathrm{r}_{ij}\right\} = 0 \ . \tag{5.20}$$

Eq. (5.20) always admits one finite solution iff there exist at least one positive and one negative entry on column j of the edge matrix $[\mathrm{r}_{ij}]_{m\times m}$. (See Appendix B for details on the solution of (5.20) and its regularized version that we use in some particular cases.)

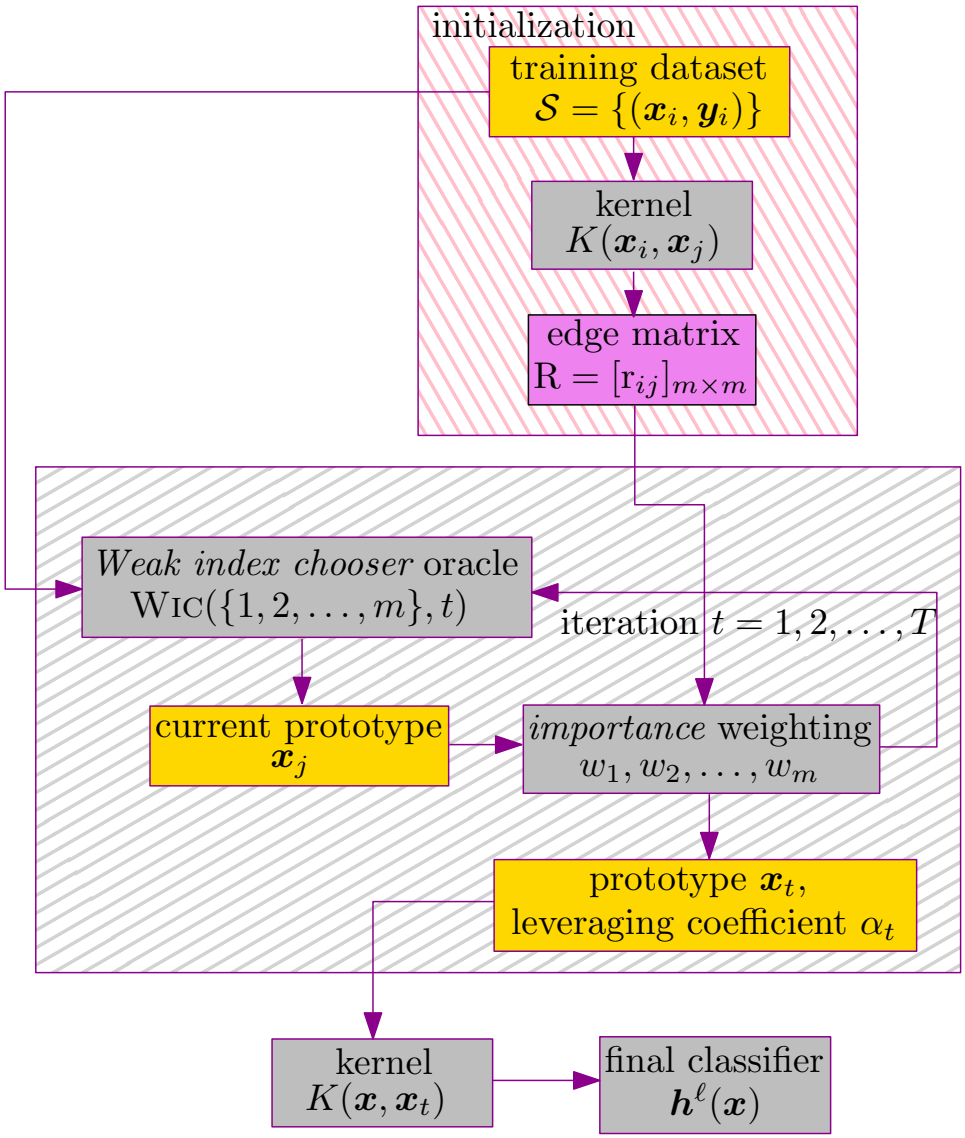


Figure 5.2: *Block diagram of the* MLNN *kernel-based learning scheme.*

Algorithm 4 MULTI-CLASS LEVERAGED k-NN MLNN ($\mathcal{S}$)

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

Let

$$\mathrm{r}_{ij} \doteq \frac{1}{C} \sum_{c=1}^{C} K(\boldsymbol{x_i}, \boldsymbol{x_j}) y_{ic} y_{jc} \tag{5.21}$$

Let $\alpha_j \leftarrow 0, \quad \forall j = 1, 2, ..., m$

Let $w_i \leftarrow 1/m, \quad \forall i = 1, 2, ..., m$

for $t = 1, 2, ..., T$ **do**

[I.0] Weak index chooser oracle:

Let $j \leftarrow \text{WIC}(\{1, 2, ..., m\}, t)$

[I.1] Compute δ_j solution of:

$$\sum_{i=1}^{m} w_i \mathrm{r}_{ij} \exp\{-\delta_j \mathrm{r}_{ij}\} = 0\,; \tag{5.22}$$

[I.2] Let

$$w_i \leftarrow w_i \exp(-\delta_j \mathrm{r}_{ij}), \ \forall i : \mathrm{r}_{ij} \neq 0\,; \tag{5.23}$$

[I.3] Let $\alpha_j \leftarrow \alpha_j + \delta_j$

Output: $h_c^{\ell}(\boldsymbol{x}_q) = \sum_{t=1}^{T} \alpha_j K(\boldsymbol{x}_q, \boldsymbol{x}_j) y_{jc}\,, \quad \forall c = 1, 2, \ldots, C$

5.2.4 MLNN: Multi-class Leveraged k-NN rule

A block diagram of our MLNN algorithm is displayed in Fig. 5.2, where the initialization phase, *i.e.*, computing the kernel and the edge matrix on training data, is clearly distinguished from the training phase, which consists of the iterative "re-weighting" of the annotated examples and the construction of the prototype set ("importance weighting"). Finally, the classification phase involves computing the similarity kernel between a new observation and all prototypes and applying the multi-class classification rule.

Pseudocode of MLNN is shown in Alg. 4. As it is common to boosting algorithms, MLNN operates on a set of weights w_i ($i = 1, 2, ..., m$) defined over training data. Weights are repeatedly updated. At each iteration t of the algorithm, a *weak index chooser* oracle $\text{WIC}(\{1, 2, ..., m\}, t)$ determines index $j \in \{1, 2, ..., m\}$ of the example to leverage (step I.0). Various choices are possible for this oracle. The simplest is perhaps to compute Eq. (5.22) for all the training examples. Indeed, δ_j in Eq. (5.22) can be used to obtain a local measure of the class density [NN09c], which is as better as δ_j gets large. This simple oracle thus picks j maximizing δ_j:

$$j \leftarrow \text{WIC}(\{1, 2, ..., m\}, t) \ : \ \delta_j = \max_{j \in \{1, 2, ..., m\}} \delta_j^t\,. \tag{5.24}$$

This oracle allows an example to be chosen more than once, thus letting its leveraging coefficient α_j be updated several times (step I.3). It is known that, in order to be statistically consistent some boosting algorithms require to be run for $T \ll m$ rounds [BJM06]. Cast in the setting of MLNN, this constraint precisely supports prototype selection, as T is an upperbound for the number of examples with non-zero leveraging coefficients.

The main ingredient to compute leveraging coefficients relies on the *averaged edge matrix* R with general term $\mathrm{r}_{i,j}$ (Eq. 5.19). This generally depends on the pairwise similarity between two training examples, as it is given by the kernel. In the following, we describe the three main implementations of MLNN, depending on which kernel is considered.

5.2.5 Kernels for MLNN

k-NN kernel In the most basic setting, that is using the k-NN kernel, term $K(\boldsymbol{x_i}, \boldsymbol{x_j})$ in (5.19) behaves like a "hard windowing" function that only selects those examples j that are k-NN of i. In this case (5.19) simplifies to:

$$\mathrm{r}_{ij} \doteq \begin{cases} \frac{1}{C} \sum_{c=1}^{C} y_{ic} y_{jc} & \text{if} \quad \boldsymbol{x}_j \in \mathrm{NN}_k(\boldsymbol{x}_i) \\ 0 & \text{otherwise} \end{cases} \tag{5.25}$$

and (5.22) has the following closed-form solution:

$$\delta_j \leftarrow \frac{(C-1)^2}{C} \log \left(\frac{(C-1) w_j^+}{w_j^-} \right) , \tag{5.26}$$

with:

$$w_j^+ = \sum_{i:\, \mathrm{r}_{ij} > 0} w_i, \quad w_j^- = \sum_{i:\, \mathrm{r}_{ij} < 0} w_i \, . \tag{5.27}$$

Notice that, when whichever w_j^+ or w_j^- is zero, δ_j in (5.26) is not finite. There is a simple way to eliminate this drawback, inspired by [SS99]: we add $1/m$ to both the numerator and the denominator of the fraction in the log term of (5.22). This smoothes out δ_j, thus guaranteeing its finiteness without impairing MLNN convergence. This correction is part of a more general regularization of Eq. 5.22, which is critical when using any truncated kernel, as in this case the solution is not guaranteed to be finite. (See discussion in Appendix B.)

Generic kernel When using a generic kernel, entries of the edge matrix (Eq. 5.19) are real-valued, thus the solution of transcendental equation (5.20) has to be computed numerically. In this thesis, we propose to solve (5.20) by a Newton's iterative method, which gives the following approximation at step $k+1$, given the previous one at step k:

$$\delta^{(k+1)} = \delta^{(k)} + \frac{\sum_{i=1}^{m} w_i \mathrm{r}_{ij} \exp\left\{-\delta^{(k)} \mathrm{r}_{ij}\right\}}{\sum_{i=1}^{m} w_i \mathrm{r}_{ij}^2 \exp\left\{-\delta^{(k)} \mathrm{r}_{ij}\right\}} \, . \tag{5.28}$$

A crucial setting for obtaining quick convergence to the solution is the initialization. Namely, we propose to initialize the algorithm with the root of a linearized version of Eq. (5.22):

$$\delta^{(0)} = \frac{\sum_{i=1}^{m} w_i \mathrm{r}_{ij}}{\sum_{i=1}^{m} w_i \mathrm{r}_{ij}^2} \,. \tag{5.29}$$

In our experiments, we tested two implementations using, respectively, a *Radial Basis Function* (RBF) kernel and a *Histogram Intersection kernel*. The first kernel is generally defined as a Gaussian, which provides "smooth" pairwise similarities between feature points:

$$K_{\mathrm{RBF}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left\{-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}{2\sigma^2}\right\} \,, \tag{5.30}$$

where parameter σ may be either constant or adapted to the local sample density (*i.e.*, $\sigma = \rho_k(\boldsymbol{x_i})$, the k-NN Euclidean distance from $\boldsymbol{x_i}$). In this chapter, we consider a Gaussian kernel that is truncated to the first k nearest neighbors, thus providing a straightforward generalization of the k-NN boosting. In this case the edge matrix writes as follows:

$$\mathrm{r}_{ij} \doteq \begin{cases} \frac{1}{C}\sum_{c=1}^{C} \exp\left\{-\frac{||\boldsymbol{x_i}-\boldsymbol{x_j}||^2}{2\sigma^2}\right\} y_{ic} y_{jc} & \text{if} \quad \boldsymbol{x}_j \in \mathrm{NN}_k(\boldsymbol{x}_i) \\ 0 & \text{otherwise} \end{cases} \,. \tag{5.31}$$

The latter kernel, *i.e.*, the Histogram Intersection kernel, is particularly adapted to histogram-like feature vectors and is generally defined as follows [Cha08a]:

$$K_{\mathrm{HI}}(\boldsymbol{x_i}, \boldsymbol{x_j}) = \sum_{h=1}^{d} \min\left(x_{ih}, x_{jh}\right) = \frac{1}{2}\sum_{h=1}^{d}\left[x_{ih} + x_{jh} - |x_{ih} - x_{jh}|\right] \,. \tag{5.32}$$

This definition of the Intersection kernel is strongly related to the Manhattan (ℓ_1) distance for "true" histograms, *i.e.*, ℓ_1-normalized histograms (denoted by $\tilde{\boldsymbol{x}}$). Indeed, it is simple to verify that the following relationship holds in this case:

$$K_{\mathrm{HI}}(\tilde{\boldsymbol{x}}_{\boldsymbol{i}}, \tilde{\boldsymbol{x}}_{\boldsymbol{j}}) = 1 - \frac{1}{2}||\boldsymbol{x}_i - \boldsymbol{x}_j||_1 \,. \tag{5.33}$$

We exploit this relationship for improving time efficiency of computing the egde matrix induced by the kernel (5.32). Namely, similarly as for the Gaussian RBF kernel defined above, we propose to truncate the Histogram Intersection kernel (5.32), only keeping the k largest similarity values and setting the others to zero. Due to (5.33), this is equivalent to searching for k-NN wrt the Manhattan distance, thus specializing the edge matrix as follows:

$$\mathrm{r}_{ij} \doteq \left[\frac{1}{C}\sum_{c=1}^{C} y_{ic} y_{jc}\right] \cdot \left[\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||_1}{\rho_k^1} < 1\right] \cdot \left[1 - \frac{1}{2}||\boldsymbol{x}_i - \boldsymbol{x}_j||_1\right] \,, \tag{5.34}$$

where are clearly distinguished three factors: (1) the usual label-dependent term, (2) the selection of k-NN wrt the ℓ_1 norm (ρ_k^1 is the k-NN Manhattan distance), which can be efficiently performed using tailored data structures like kD-trees, and (3) a linear weighting only depending on the Manhattan distances.

The different solutions of Eq. 5.22 depending on whether a "smooth" or a "truncated" kernel is used are described in Appendix B with more details.

5.2.6 Properties of MLNN

In this section we provide formal details about our boosting analysis of MLNN. In particular, we give the statements of two fundamental theorems for MLNN, which bring to the multi-class case the same theoretical properties that hold for UNN, as stated in Sec. 4.2.5. Indeed, the first theorem states the convergence of MLNN to the global minimum of the exponential risk.

Theorem 3. MLNN *converges with T to $\boldsymbol{h}^\ell$ realizing the **global** minimum of the exponential risk (5.13).*

In Appendix B.1 we prove this theorem by giving the necessary condition for Eq. 5.22 to admit a finite solution. Furthermore, as done for UNN in Chap. 4, we provide the general learning algorithm for the minimization of a broad class of multi-class surrogate risk [BJM06, NN09c].

The second theorem provides a convergence rate for MLNN, which is based on a fundamental assumption on weak classifiers.

Theorem 4. *Let $p_j \doteq w_j^+ / (w_j^+ + w_j^-)$ and $||\boldsymbol{w}||_1 = \sum_{i=1}^m w_i$. If the following weak index assumption (**WIA**) holds for $\tau \leq T$ steps in* MLNN*:*

(WIA) *There exist some $\gamma > 0$ and $\eta > 0$ such that the following two inequalities hold for index j returned by* $\textsc{Wic}(\{1, 2, ..., m\}, t)$*:*

$$|p_j - 1/\mathrm{C}| \geq \gamma \ , \tag{5.35}$$

$$(w_j^+ + w_j^-)/||\boldsymbol{w}||_1 \geq \eta \ . \tag{5.36}$$

Then: $\varepsilon^{0/1}(\boldsymbol{h}^\ell, \mathcal{S}) \leq \exp(-\frac{C}{C-1}\eta\gamma^2\tau)$.

Ineq. (5.35) is the usual weak learning assumption, used to analyze classical boosting algorithms [SS99], when considering examples as weak classifiers. A *weak coverage assumption* (5.36) is needed as well, because insufficient *coverage* of the reciprocal neighbors could easily wipe out the surrogate risk reduction due to a large γ in (5.35). In the framework of k-NN classification, 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 $\varepsilon^{\exp}(\boldsymbol{h}^\ell, \mathcal{S})$. This is important, as a big difference with classical

boosting algorithms (*e.g.*, AdaBoost [FS95]) is that oracle $\textsc{Wic}(.,.)$ has access only to m different weak classifiers. Finally, the bound in Theorem 4 shows that classification (5.35) may be more important than coverage (5.36) for nearest neighbors.

5.2.7 MLNN and SVM

The formulation of our MLNN classification rule (5.12) is very similar to that of support vector machines (SVM) for the binary case. Indeed, for $C = 2$, (5.12) is a kernel-based classifier that linearly separates positive and negative data using a sparse set of prototypes. These prototypes are analogous to *support vectors* of SVM, as they are a subset of the learning data that determine the decision boundary. Furthermore, as support vectors' coefficients are related to their distance to this boundary, similarly our leveraging coefficients α_j represent the "prototypical relevance" of data. However, the relationship between weighting coefficients and the distance to the decision boundary is different for SVM than MLNN. Indeed, SVM learning favours the selection of data that are located close to the boundary as support vectors, whereas our MLNN approach tends to give larger leveraging coefficients to the prototypes located further from the boundary, where the class conditional probability is higher. Similarly, while data that are outside the margin region are not useful to determine the decision boundary of SVM, thus being rejected, conversely in MLNN adding prototypes that are close to the boundary does not impact on decreasing the risk function significantly, due to their small edges. This is mainly due to the different formulations of the two learning algorithms as optimization problems. On the one hand, SVM minimize the *hinge loss* over training data with a quadratic regularizer, which constrains the maximum ℓ_2-norm of support vectors' coefficients [Bis07]. On the other hand, MLNN minimizes a boosting-like strictly convex risk, *e.g.*, the exponential risk, with a sparsity inducing constraint on the ℓ_1-norm of the leveraging coefficients, which is controlled by setting the number of boosting iterations T [RSMM00]. A comparison of both SVM and MLNN loss functions is shown in Fig. 5.3, as well as the true empirical loss.

The difference between learning using SVM and MLNN is clearly displayed in Fig. 5.4, 5.5, where the two methods are applied to the Ripley's dataset. In the first figure, we compare linear SVM with our basic MLNN implementation (k-NN kernel), whereas in the second figure kernel SVM and kernel MLNN are compared. Namely, we set the same kernel parameters for the two methods, *i.e.*, Gaussian kernel with $\sigma = \frac{1}{\sqrt{2}}$ and we set the constraint parameters (C parameter of SVM and number of boosting rounds T of MLNN) so as to retain the same number of support vectors (top figure) as prototypes (bottom). Under these conditions, both methods are able to learn a good boundary between the two classes, although

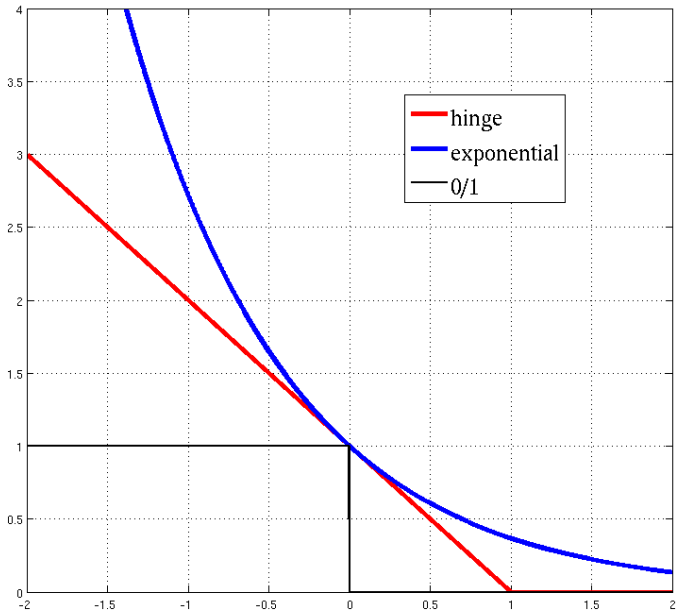
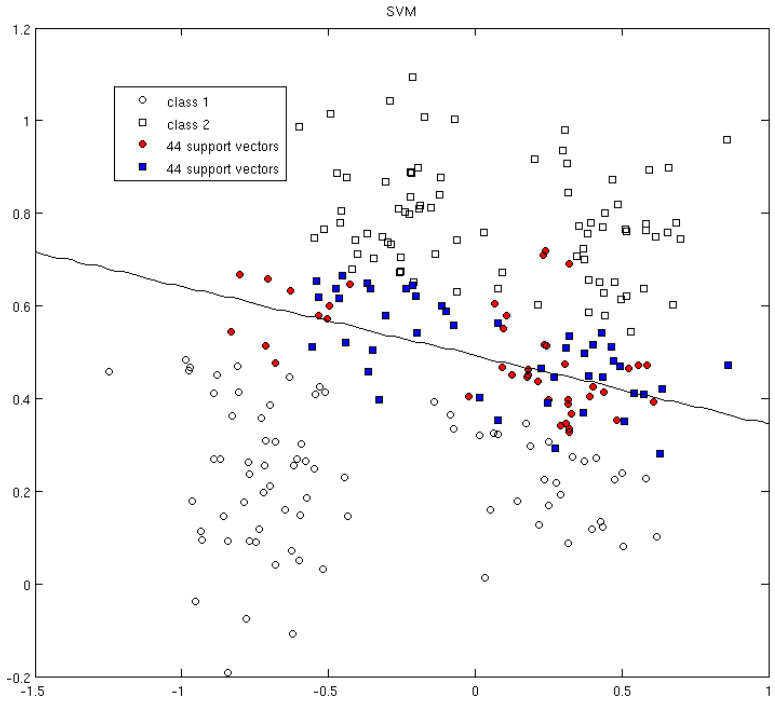


Figure 5.3: *Comparison of SVM (*hinge*) and* MLNN *(*exponential*) loss functions with the true empirical (*0/1*) loss.*

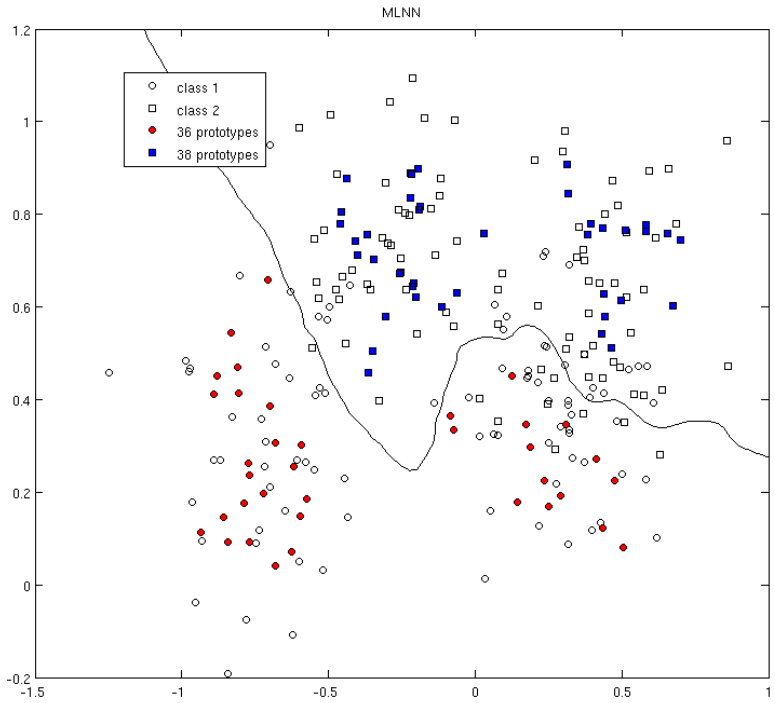
the SVM boundary is a bit "smoother" that that of MLNN. However, the main difference lies in the position of the selected data that are retained for classification. Indeed, while most of the support vectors selected by SVM are very close to the boundary, MLNN prototypes are mostly located around the modes of the class conditional distributions. Roughly speaking, this simple synthetic example shows that, while support vectors aim at inducing the decision boundary directly, our MLNN prototypes are instead more directly related to a sparse representation of the class distributions, thus inducing the boundary indirectly. In this sense, our technique is more "generative" than a purely discriminative approach like SVM, and thus is more easily amenable to treating multiple classes.

§ 5.3 Experiments

In this section, we present experimental results of MLNN with different kernel settings and compare them with both baseline k-NN and some state-of-the-art machine learning methods, like ITML [DKJ+07] and SVM [CST00]. In most of our experiments, we focused on evaluating how the average classification precision varies as a function of the number of prototypes that are used for testing. Indeed, one of the main features of our method is to allow to explicitely fix the number of data to be used at classification time, thus directly bounding the computational cost of the test phase. In

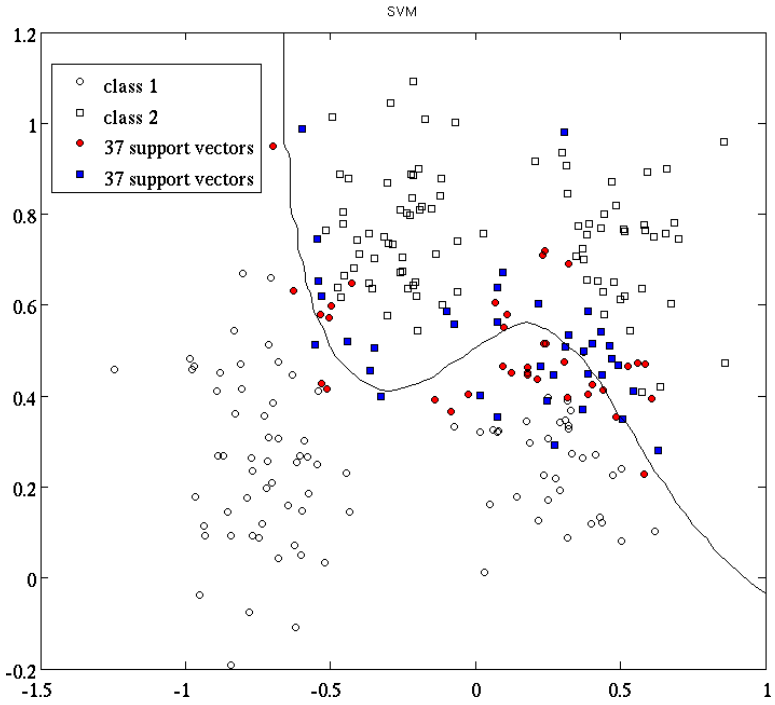


(a)

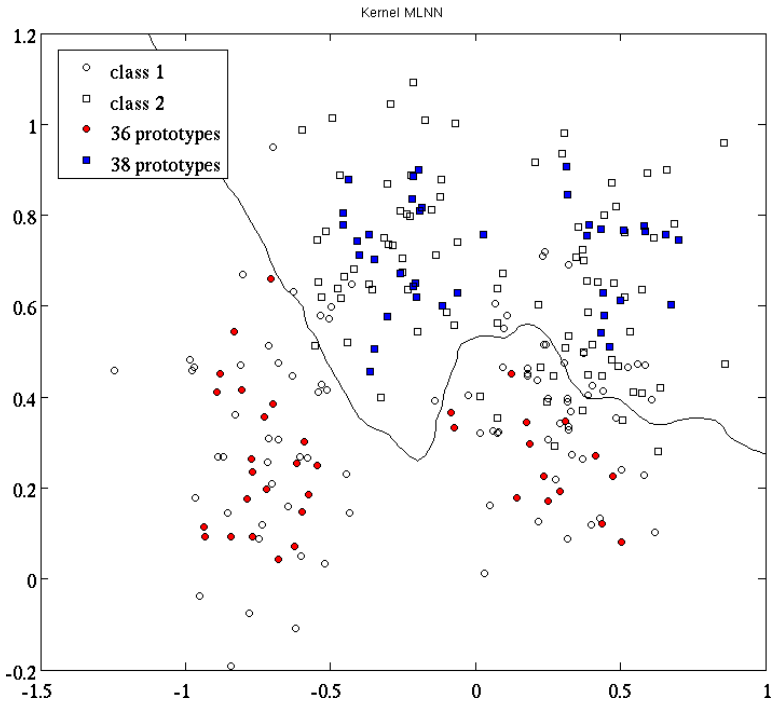


(b)

Figure 5.4: *Classification boundary on the Ripley's dataset for: (a) linear SVM with SVM cost parameter* $C = 10^3$*; (b)* MLNN *with* $k = 5$ *and* $T = 74$*. Notice that our method with the most basic setting (*k*-*NN *kernel) is able to learn a good classification boundary compared to linear SVM.*



(a)



(b)

Figure 5.5: *Classification boundary on the Ripley's dataset for: (a) SVM using Gaussian kernel (5.31) with* $\sigma = \frac{1}{\sqrt{2}}$ *and SVM cost parameter* $C = 10^3$*; (b)* MLNN *with the same kernel settings and* $T = 74$*.*

particular, in all the reported experiments we carried out prototype selection by setting $T < m$, which corresponds to retaining at most T relevant prototypes (the exact number of prototypes depends on which criterion is chosen for the WIC oracle, namely whether allowing a prototype to be selected at multiple steps or not). When running the baseline k-NN method, we carried out *random prototype selection*, which is the easiest strategy for data reduction, and averaged the classification results over a number of iterations.

In the following sections, we report image categorization results on the following three datasets, containing 8, 13 and 15 categories of real-world images, respectively:

8-cat: firstly proposed by [OT01], includes 2,688 color images grouped into eight categories: 360 coast, 328 forest, 374 mountain, 410 open country, 260 highway, 308 inside of cities, 356 tall buildings, and 292 street.

13-cat: adds five more categories of gray-scale images to the 8-cat database [FFP05]: 241 suburb residence, 174 bedroom, 151 kitchen, 289 living room, and 216 office.

15-cat: includes 13-cat database plus two more categories (gray-scale images) [LSP06]: 315 store, and 311 industrial.

In order to represent these images in terms of feature sets, we tested the two most common descriptors for natural image classification, *i.e.*, Gist features [OT01] and Bag-of-Features histograms computed from SIFT descriptors [SZ03]

This section is organized by first reporting results of natural image categorization relying on global Gist descriptors (Sec. 5.3.1), where the benefits of using our classification method over classic k-NN-based methods are shown in terms of both classification precision and computational cost. Then, we focus on classification using the state-of-the-art Bag-of-Features descriptors, by investigating the most suitable dissimilarity measures for such descriptors and comparing our method with SVM (Sec. 5.3.2).

5.3.1 Scene categorization using Gist descriptors

Settings In this section we report results obtained by splitting each of the considered databases in two distinct subsets, one for training, the other for test. Following the settings of [BZM08], we always used about 2,000 randomly selected training images. Namely, 250 images per category were selected from 8-cat database, 150 from 13-cat and 15-cat. The remaining images were used to measure classification performances. In our experiments, we mostly concentrated on evaluating the benefit provided by selecting a *sparse* prototype dataset from training data. For this purpose, all the results of MLNN are compared with those of classic k-NN for a fixed number of

prototypes, *i.e.*, for a fixed computational cost of classification. (So as for k-NN, a random sample of the training set was selected and results were averaged over a number of random sampling realizations.) All the results we present were obtained with $k = 11$ and pre-processing Gist features with PCA down to dimension $d = 128$, which provided the best results.

We also compared our MLNN algorithm with several methods on the 8-cat database. On the one hand, we tested:

- MLNN with the basic setting, *i.e.*, the uniform k-NN kernel of Eq. (5.25);
- MLNN with fixed-size Gaussian kernel, which we denote as WMLNN; namely, we set $\sigma = 0.25$ in kernel (5.30);
- MLNN with adaptive-size Gaussian kernel, called AdaWMLNN, for which we set $\sigma = \sqrt{2}\rho_k(\boldsymbol{x}_i)$ in kernel (5.30) ($\rho_k(\boldsymbol{x}_i)$ being the k-NN distance from example $\boldsymbol{x}_i$);
- MLNN "one-versus-all", *i.e.* Alg. 1 with $C = 2$ applied to each category independently (considering examples in the current category as "positives", the remaining ones as "negatives").

Furthermore, we compared our method with different k-NN–based classification methods, which either rely on metric learning or not. Namely, we tested:

- classic non-parametric k-NN;
- weighted k-NN (Wk-NN) voting with Gaussian weights, as proposed by Philbin et al. [PCI+08]; namely, we use weighting factor (5.30) with $\sigma = 1$;
- k-NN voting with ITML metric learning [DKJ+07].

We tested all these methods for a fixed number of prototypes, *i.e.*, for a fixed computational cost of classification. In particular, a random sample of the training set was selected and results were averaged over a number of random sampling realizations.

Finally, we integrated ITML metric learning with MLNN in order to provide a unique method for *simultaneously* addressing the choice of the metric distance and the rejection of "noisy" examples, which are the two fundamental issues to be addressed for improving over k-NN classification.

Results The categorization test consists in assigning each test image to one of the predefined categories. We measured the overall performance rate as the mean Average Precision (MAP), which is the average of the classification rates for each category. The results reported in Fig. 5.6(a) show the significant improvement provided by using a "smooth" kernel for learning

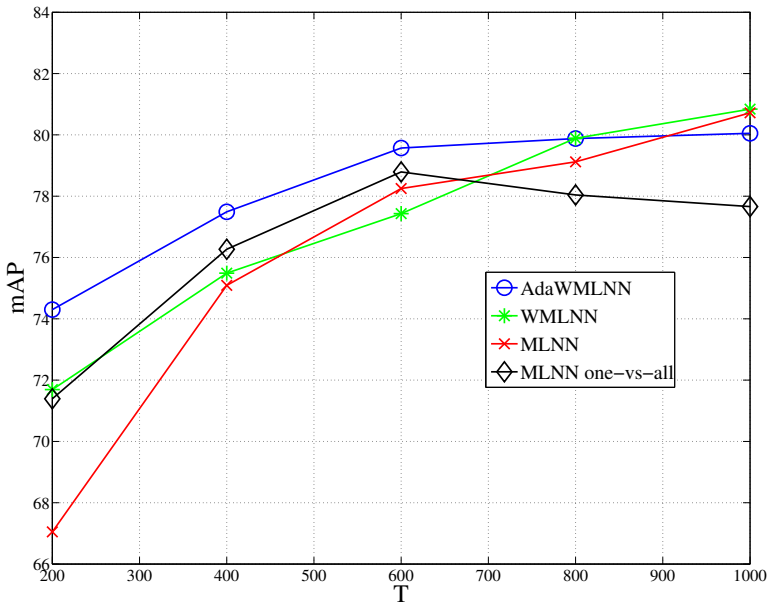
the prototypes. Namely, the adaptive-size kernel provides the best performances. Furthermore, the gap over the basic MLNN is more consistent when retaining less prototypes, as AdaWMLNN it enables a finer class density estimation even with very sparse examples (see, for instance, the performance gap of 7% between MLNN and AdaWMLNN for $T = 200$). Finally, notice that the multi-class version of our algorithm outperforms the one-versus-all implementation (gap between 1% and 3%). Hence, our multiclass MLNN not only is much less computationally expensive than one-versus-all MLNN (which is equivalent to UNN described in Chap. 4), as it avoids to run the boosting procedure C times independently, but it also provides better classification accuracy.

On the same 8-cat database we also compared AdaWMLNN to k-NN voting with or without metric learning (Fig. 5.6(b)). First of all, we notice that our AdaWMLNN method significantly outperforms k-NN and Wk-NN, *i.e.*, non-learned voting rules (up to 6% improvement). Then, performances of our method are overall comparable to those of ITML, being slightly inferior to them, but the computational cost of MLNN is considerably lower than that of metric learning. Finally, our results show that, when combined with a metric learning strategy, MLNN is able to significantly outperform all the other classification methods, thus enabling a significant accuracy improvement over the state-of-the-art (up to 3% when retaining few prototypes, *e.g.*, see performance at $T = 200$).

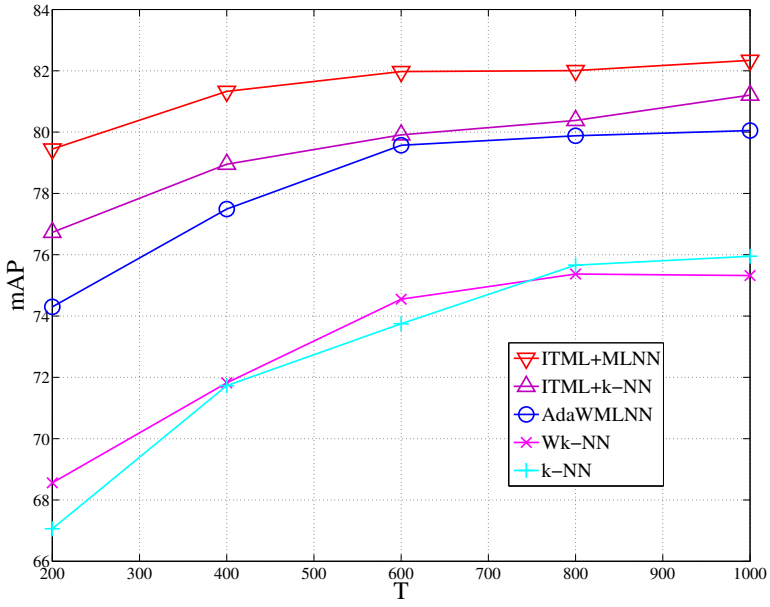
In Fig. 5.9 we show an example of the 100 most relevant prototypes retained when running MLNN on 8-cat database. Intuitively, the number of prototypes per category is automatically adapted to better represent categories with larger intra-class variability (mostly in terms of texture information, *e.g.*, compare categories "tall building" and "coast").

We also evaluated the confusion table (Fig. 5.7), which highlights the difficulty of discriminating between couples of visually similar image categories. *E.g.*, images of "tall building" are often confused with "inside city", "mountain" with "open country". In spite of this phenomenon, which is mainly due to ambiguous ground-truth annotation and ill-defined image categories and has been already reported on this database [VS07], performances of MLNN are overall satisfactory and comparable to state-of-the-art. In Tab. 5.1 we report the best results of both our method and regular k-NN on the three databases. We observe that the improvement over unsupervised classification becomes more consistent when dealing with more challenging categories, like 15-cat database. (Remark the gap of 3.5% in terms of MAP.) At the same time, MLNN considerably reduces the computational cost of image classification. *E.g.*, on 15-cat database, the best precision of MLNN is obtained by decreasing the prototype number by a factor 2.5 (from 150 to 60).

In Fig. 5.8 we focus on a more extensive comparison between regular MLNN and classic k-NN on the 13-cat and 15-cat datasets. Here, we report



(a)



(b)

Figure 5.6: *Experimental results of categorization on 8-cat database in terms of MAP as a function of the number of prototypes, for* $k = 11$ *and Gist descriptors of dimension* $d = 128$ *(after PCA). (a) Comparison between 3 different implementations of* MLNN *and one-versus-all* MLNN*. (b) Comparison between* MLNN *with adaptive Gaussian kernel (AdaWMLNN), k-NN, weighted k-NN and* MLNN *one-versus-all (UNN).*

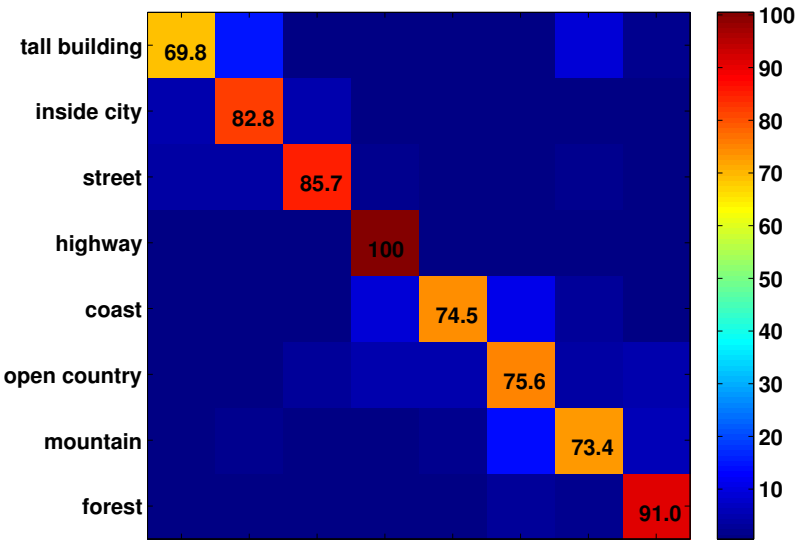
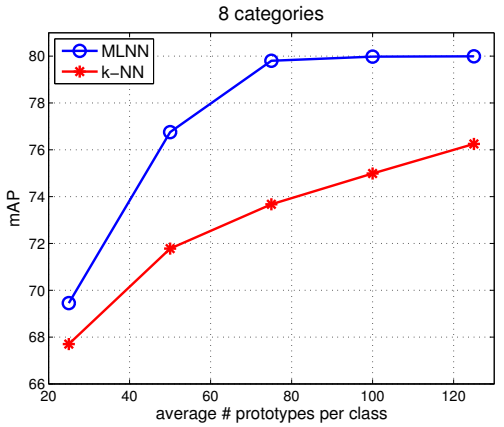


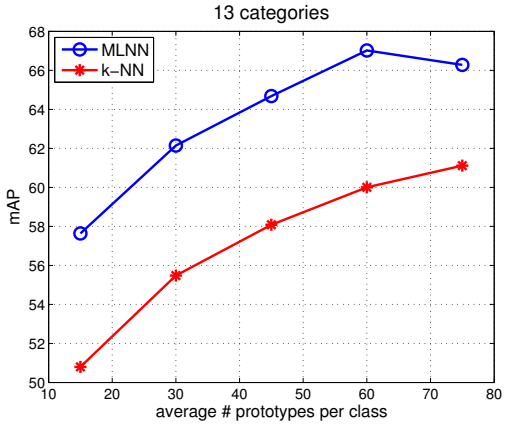
Figure 5.7: *Confusion table for* MLNN *on the 8-cat database.*

Table 5.1: *Performance comparison as a function of the average number of prototypes per class.*

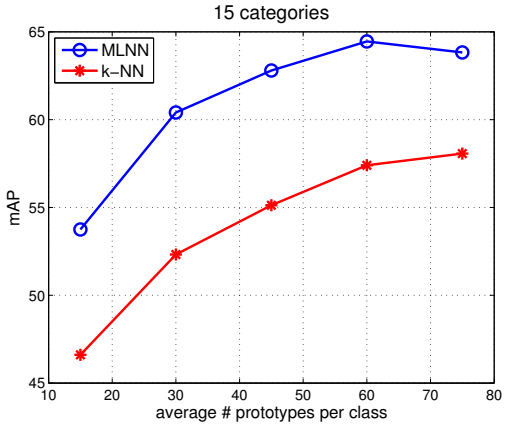
		# prototypes	MAP
8 cat	MLNN	175	81.0
	k-NN	250	79.2
13 cat	MLNN	60	67.0
	k-NN	150	65.6
15 cat	MLNN	60	**64.5**
	k-NN	150	**61.0**



(a)



(b)



(c)

Figure 5.8: *Performance of Gist-based classification using* MLNN *compared to k-NN as a function of the average number of prototypes per class on (a) 8-cat, (b) 13-cat and (b) 15-cat datasets.*

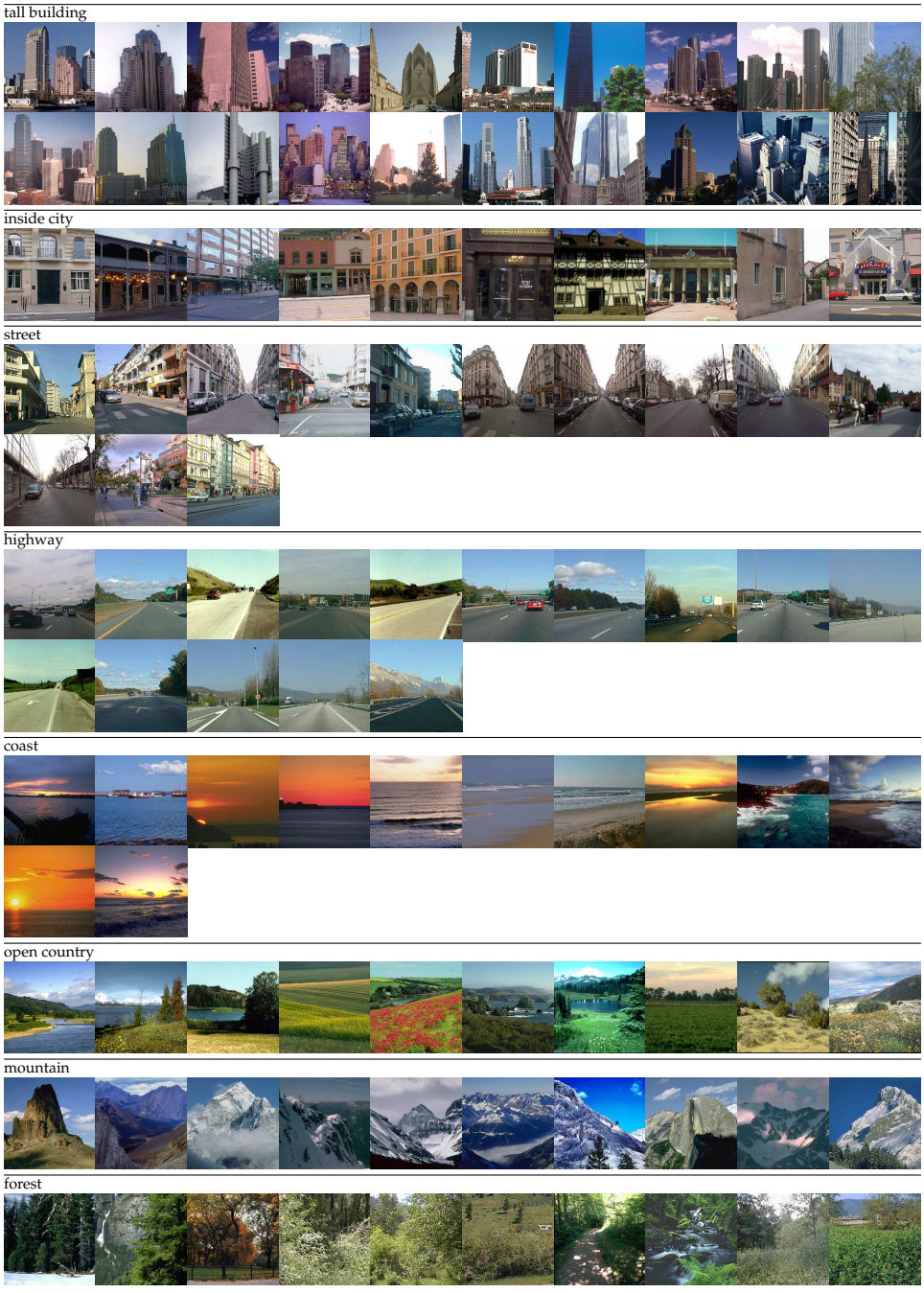


Figure 5.9: *100 prototypes selected by* MLNN *among 2,000 training images of the 8-categories database.*

the Mean Average Precision (MAP) as a function of the number of prototypes per category. (Since this number varies from category to category, we report the average number of prototypes over all categories.) Notice that the gap between the two methods is most significant when retaining less than half prototypes, namely 6% improvement with 80 prototypes on 8-cat database, 7% with only 60 prototypes for both 13-cat and 15-cat. Besides considerably improving precision over k-NN, we also drastically reduce the computational complexity of classification, which deals with finding nearest neighbors on a sparse dataset (gain up to a factor 4 when discarding half prototypes).

We also tested our method for *ranking*, thus investigating its behaviour when dealing with confusing or ill-defined image categories, as proposed by Vogel and Schiele [VS07]. In particular, whereas the categorization test allocates an image to the category with the maximum score (Sec. 5.2.1), the ranking test considers the first r categories with the largest scores, thus virtually giving multiple labels to the image. Tab. 5.2 compares categorization rates (the two columns for $r = 1$) with the ranking classification rates when using the best label as well as the second one (columns for $r = 2$). The results, which are given for the 15-cat database, clearly show that MLNN significantly benefits from this setting on the most challenging categories. The large precision jump on images of such categories (highlighted in the table) suggests that those images are likely to be very close to those of another category in the feature space (e.g., "tall building" and "inside city", "coast" and "mountain"). Indeed, visual descriptors are not sufficient to convey relevant information for the semantical categorization of such images. Overall performance (last line in the table) jumps from 64.5% to 80.7% (more than 16% improvement), while keeping a significant gap over k-NN (more than 5%).

5.3.2 Categorization using Bag-of-Features descriptors

Settings In the context of generic image categorization, the *Bag-of-features* scheme is among the best performing feature representation methods. Besides its simplicity, the main advantage of this image representation approach is the *one-to-one* association between images and feature vectors, that allows for a straightforward use of discriminative learning tools like k-NN and SVM. Originally proposed for text categorization [SM83], the Bag-of-Features descriptors have been successfully applied to image classification problems, with several implementations, ranging from using them as feature vectors for discriminative learning [CDF+04] to more sophisticated approaches like pyramid match kernels [GD05, LSP06].

In this section we present and discuss results we obtained using MLNN for Bag-of-Features classification on the most challenging of the three databases described in Sec. 5.3, *i.e.*, the 15-cat database, which mixes outdoor scenes

Table 5.2: *Rank statistics for* MLNN *using Gist descriptors on the 15-categories database.*

	MLNN		k-NN	
category	$r = 1$	$r = 2$	$r = 1$	$r = 2$
tall building	57.3	**71.8**	46.1	64.1
inside city	64.6	**76.6**	56.3	72.8
street	81.0	90.1	77.5	88.7
highway	78.2	88.2	80.0	86.4
coast	76.2	**91.0**	60.5	86.7
open country	57.7	78.1	68.1	81.5
mountain	59.8	**88.8**	35.7	73.2
forest	86.5	91.0	85.4	88.8
suburb	85.7	91.2	91.2	95.6
bedroom	57.6	75.8	57.6	74.2
kitchen	45.0	**73.3**	33.3	58.3
living room	37.4	**67.6**	40.3	61.9
office	69.2	**87.7**	43.1	76.9
industrial	40.4	**59.6**	31.1	45.3
store	70.3	80.0	62.4	78.2
MAP	64.5	**80.7**	57.9	75.5

with more difficult indoor scenes. All the results presented in this section refer to 10-fold *cross-validation*. Thus we split the 15-cat database in 10 distinct random folds, in order to form 10 different training/test subsets, each one containing more than 4,000 training images and about 450 testing images. The following statistics refer to averaging over these ten folds. For each training/test combination we first built a *vocabulary* of visual words, that are SIFT descriptors densely extracted at four resolution levels on a fixed regular grid (typically 1,000÷10,000 SIFT per image). In order to build the visual vocabulary, we ran k-means with $k = 1500$, thus representing each image as a feature vector in dimension $d = 1500$.

Histogram Intersection metrics Although BoF descriptors are widely used in most state-of-the-art image classification techniques, some crucial issues are still unsolved and may significantly impact on classification performances. In particular, we consider the two following problems:

1. how to *normalize* such image descriptors in order to make comparison between different images as unbiased as possible;

2. which *distance metric* to use for measuring the dissimilarity between two descriptors.

Such problems mainly arise from the histogram-based nature of BoF descriptors and have been the object of much research effort in the computer

vision community in the recent years. Indeed, on the one hand, their *normalization* is particularly crucial when images differ significantly from each other in terms of the local descriptors counts, thus resulting in largely variable descriptor norms. Thus it is common to pre-process BoF descriptors such that they have equal ℓ_1 norms. Less commonly, these descriptors have been ℓ_2-normalized, mostly when normalization is part of a pre-processing technique, like the squared root (sqrt) recently proposed by Perronnin et al [PSL10]. The most common alternative to ℓ_1-normalization for BoF descriptors is represented by the TF-IDF schema, which was originally proposed in the context of text retrieval and then successfully applied to image indexing, in order to take into account the larger informative "power" of rare visual words [SZ03].

On the other hand, defining the right *dissimilarity measure* between histograms (not necessarily ℓ_1-normalized) is challenging, and the resulting behaviour often strongly depends on the application. *E.g.*, the Euclidean distance between ℓ_2-normalized descriptors [BZM08] or TF-IDF-weighted descriptors [SZ03] are still the most common choices for image classification. All the results we report in this section refer to normalizing Bag-of-Features descriptors in terms of the ℓ_1 norm and comparing them using the Manhattan distance. This choice was motivated by our evaluation of different normalization/metric combinations that we report in Tab. 5.3, which refers to 10-fold cross-validation using k-NN ($k = 10$). In particular, we tested some of the most suitable histogram distance metrics, as defined in a recent taxonomy of histogram intersection measures [Cha08a], *i.e.*, *Manhattan*, *Canberra*, *Lorentzian*, besides the baseline Euclidean distance. Furthermore, we evaluated different descriptor normalization/pre-processing methods, like the common ℓ_1, ℓ_2 and TF-IDF [SZ03], as well as the most recent *squared root* pre-processing [PSL10] and the baseline (absence of normalization). First of all, our results show that the ℓ_1 normalization always outperforms ℓ_2 and the baseline for a fixed metric (except for the Euclidean distance, for which ℓ_2-normalization is the best), and the gap is particularly significant for distances like Manhattan and Lorentzian (more than 11% over ℓ_2-normalization). The best performances are obtained for Manhattan and Lorentzian with ℓ_1-normalization, and Manhattan with TF-IDF normalization, which still outperform the squared root pre-processing strategy. So as for the distance metric choice, our results clearly show that the Euclidean distance is generally *not* the optimal choice for comparing those histogram-like descriptors, thus suggesting intersection metrics as better alternatives. (See for instance the 10% gap between Euclidean and Manhattan for ℓ_1-normalized BoF, or the 9% gap between the same two metrics when using TF-IDF normalization.)

In Fig. 5.10 we quantify the gain provided by using the Manhattan (ℓ_1) distance over the Euclidean (ℓ_2) distance for our MLNN approach. Namely, we compare MLNN using the simple k-NN kernel of Eq. 5.2 with

Table 5.3: *Comparison of k-NN classification performances using different histogram normalization criteria and intersection metrics on the 15-cat database ($k = 10$).*

metric	normalization	MAP
Euclidean	ℓ_1	54.55
Euclidean	none	57.56
Euclidean	ℓ_2	59.60
Manhattan	ℓ_1	**64.29**
Manhattan	none	62.09
Manhattan	ℓ_2	53.19
Canberra	ℓ_1	60.80
Canberra	none	59.57
Canberra	ℓ_2	59.81
Lorentzian	ℓ_1	**64.34**
Lorentzian	none	58.21
Lorentzian	ℓ_2	52.07
Manhattan	tf-idf	**64.47**
Euclidean	tf-idf	55.22
Euclidean	ℓ_2+sqrt	60.78
Euclidean	ℓ_1+sqrt	62.22

MLNN using the Histogram Intersection kernel (5.32). Results of baseline k-NN classification are also shown for both metric distances. This plot shows that the choice of the kernel/k-NN metric significantly impacts on the precision of our method, with a 10% gap between the Euclidean k-NN kernel-based implementation and the Manhattan histogram intersection kernel-based implementation.

MLNN performances In order to evaluate the classification performances of our method, we report the trend of cross-validation MAP as a function of the overall number of prototypes used for testing (Fig. 5.11). We also compare these results, which were obtained using the MLNN implementation relying on the Histogram Intersection kernel, with our one-versus-all UNN method described in Chap. 4 and the baseline k-NN classification. Our method outperforms k-NN classification significantly (gap between 6% and 8%) while reducing the prototype dataset, thus the computational cost, considerably. (See for instance the precision of MLNN for 400 prototypes, which equals that of k-NN using 2,400 prototypes, thus resulting in a dataset reduction by a factor 6.) Furthemore, the advantage of using our multiclass MLNN algorithm over its binary counterpart, UNN, is mainly concentrated at very low prototype set sizes, *e.g.*, 10% of the original set, where the multiclass learning allows for precision improvement up to 7%.

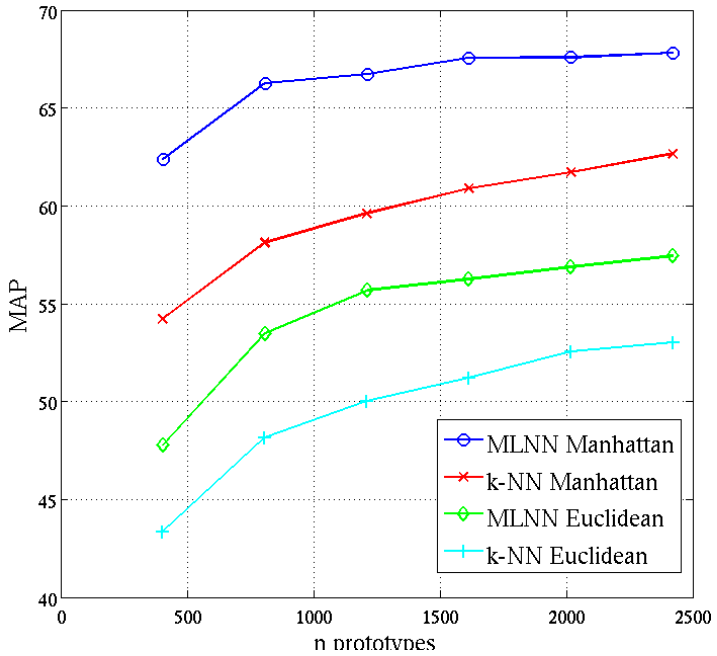


Figure 5.10: *Comparison between Manhattan (ℓ_1) and Euclidean (ℓ_2) distances for both* MLNN *and* k-NN *classification.*

Notice that the number of prototypes for UNN is reported as the average over the multiple one-versus-all problems, *i.e.*, it should be multiplied by the number of classes in order to compute the actual number of prototypes involved in classification. Hence, although UNN performances appear almost identical to those of MLNN, this latter still benefits a significant computational advantage over UNN, thus providing the best precision/cost trade-off.

Then, we look deeper into MLNN classification performances by analyzing the confusion matrix (Fig. 5.7). Overall, the method performs well on most outdoor scenes, as well as on the "office" category. Only some reasonable confusions are present, *e.g.*, between "coast" and "open country" or "inside city" and "street". Thus, average performance (MAP under 70%) drops off mainly because of the low recognition rate in a few more challenging categories, such as the indoor category "bedroom" (recognition rate of only 16.2%), and the "industrial" category (36.5%), that mixes outdoor and indoor images, thus making recognition very challenging. In particular, notice that "bedroom" images are more often misclassified into the "living room" category, due to their similar scene layouts and the presence of similar objects, *e.g.*, paintings on the walls and sofas/beds. This drastically reduces the prototypical relevance of bedroom images during the learning phase, thus biasing misclassification into the "living room" category. This

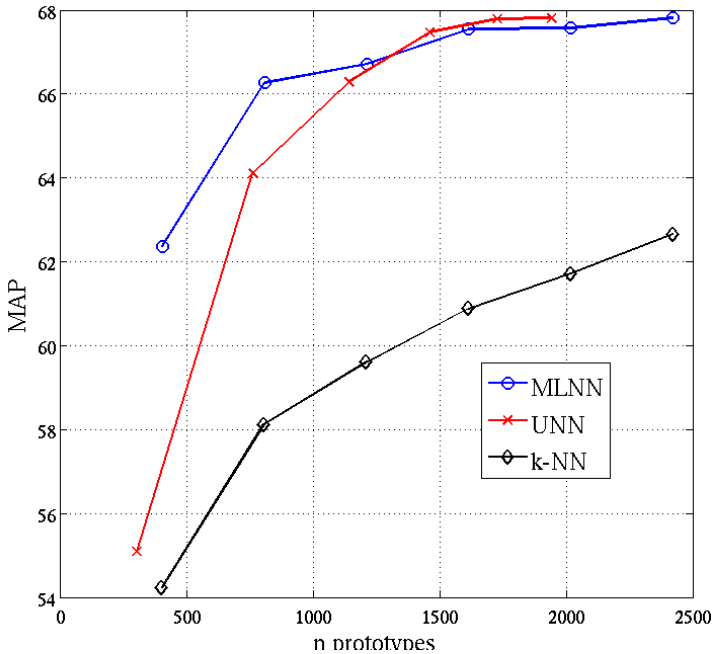


Figure 5.11: MLNN *with Histogram Intersection kernel compared to* UNN *(one-versus-all) and* k-NN. *All the three classification methods rely on the same distance metric, that is* ℓ_1 *distance.*

phenomenon is related to the well-knowm "semantic gap", which affects low-level visual descriptors that only collect statistics on the appearance information without any semantic interpretation. This problem is particularly critical for prototype-based methods when the inter-class variability is low, preventing them from learning reliable discriminating prototypes.

A very simple strategy for improving the overall precision of MLNN is to combine the scores from multiple MLNN tests, *e.g.*, by summing the classification scores corresponding to different values of k for the same (truncated) Histogram Intersection kernel. An example of the performance increase enabled by this strategy is shown in Fig. 5.13, where MLNN improves by almost 2% by summing the scores obtained for $k = 5, 10, 15$, with a best MAP of 69.23%.

Comparison with SVM Finally, we ran state-of-the-art SVM classification on the same database for the purpose of comparison, as reported in Tab. 5.4. In particular, we used the matlab implementation provided in the SVM-KM toolbox [CGGR], and carried out experiments with different settings of the RBF kernel (we report those obtained setting kernel parameter $\beta = 1$). These results show that SVM are still the best performing classification method on this dataset, with a 5% gap. Nevertheless, a major

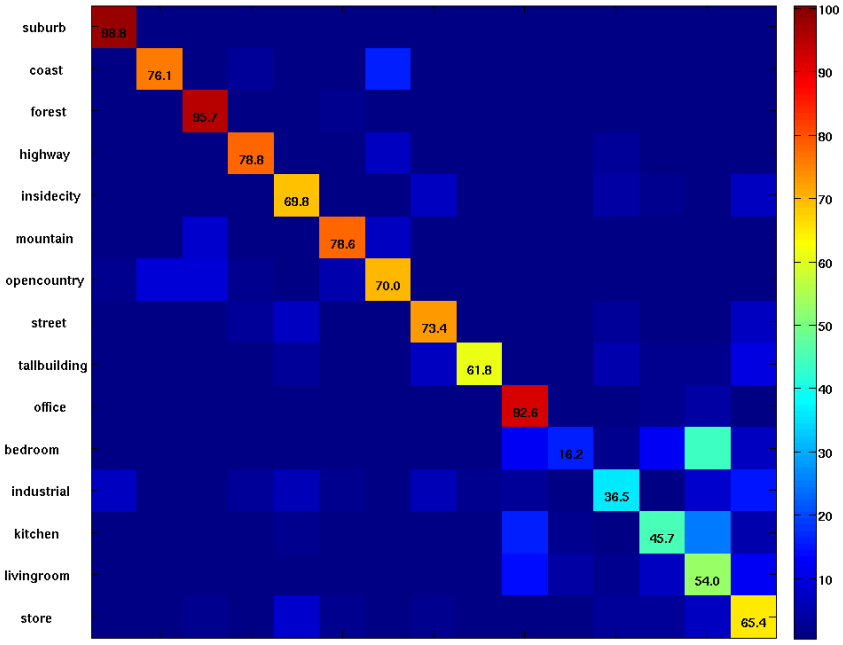


Figure 5.12: *Confusion table for* MLNN *on the 15-cat database.*

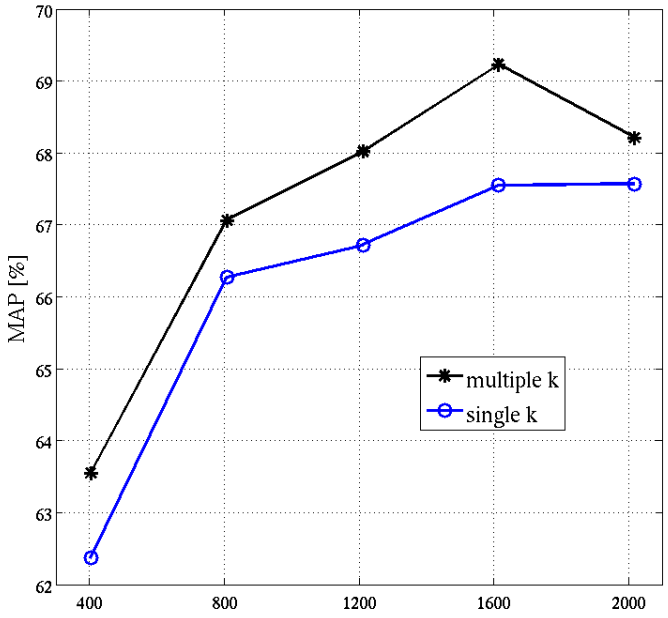


Figure 5.13: *Results of* MLNN *and k-*NN *obtained when combining multiple k values into the classification rule.*

Table 5.4: *Comparison of the best cross-validation performances for SVM,* MLNN, UNN *and* k-NN *on the 15-cat database.*

method	MAP
kernel SVM	74.47
MLNN	**69.23**
UNN	67.81
k-NN	64.29

drawback of this method is the high sensitivity of its performances to the choice of the optimization parameters (in particular, the regularization parameter C), which need to be tuned by time-consuming cross-validation. For instance, Fig. 5.14 reports the trend of MAP as a function of parameter C (in logarithmic inverse scale), showing that an inaccurate setting of this parameter makes performances decrease by 12%. Furthermore, SVM are not inherently multi-class, thus they need to treat each category separately and, since the support vectors are often far from being really sparse, the computational cost of classification, which is linear in the number of classes C and support vectors n (computational cost of order $\mathcal{O}(Cnd)$, d being the descriptor dimensionality [CST00]), becomes easily huge in real applications.

Therefore, our method, whose computational cost is $\mathcal{O}(d \log n)$ (when using kD-trees for fast k-NN search [AMN+98]) may be viewed a valuable alternative to SVM classification when computational time is an issue. Indeed, combining the simplicity of k-NN rule with the ability to build sparse prototype sets, thus reducing the computational cost while still improving the classification precision, MLNN provides a more reasonable trade-off between precision of classification and computational time for large image collections with many categories.

§ 5.4 Conclusion

In this chapter, we have tackled the problem of providing certain reasonable bounds to the classification error when using prototype-based methods like k-NN and kernel density classifiers. For this purpose, we have proposed to bring such classifiers into the boosting framework, by leveraging prototypes in order to minimize a generic (surrogate) risk function defined over training data. The algorithm we propose, MLNN, grounds on very mild hypotheses on the nature of the loss function that is used to define the surrogate risk, ranging from the classic exponential and logistic losses, very common for image classification, to more sophisticated functions that may be specifically tailored to the application at hand. Furthermore, our

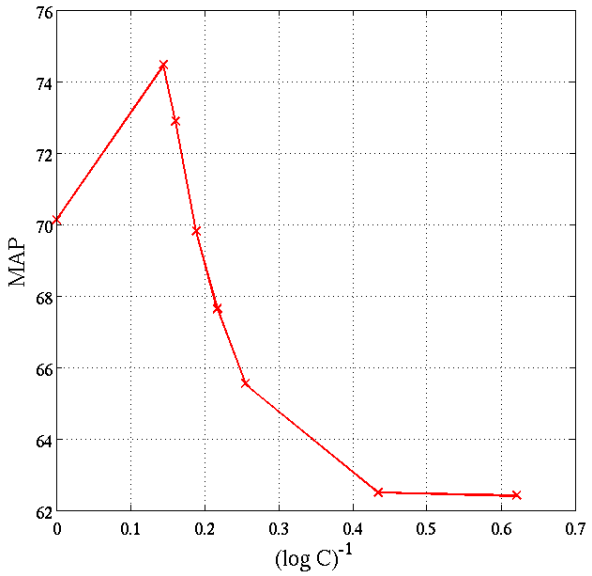


Figure 5.14: *SVM performances as a function of* $(\log C)^{-1}$*, C being the regularization parameter of SVM.*

framework adopts an inherently multi-class definition of the surrogate risk, thus really exploiting the multi-class nature of prototype voting rules and avoiding to split a single classification problem into multiple binary problems according to the widely used one-versus-all strategy. Our MLNN training phase is implemented as an iterative algorithm that, at each step, adds a new relevant prototype in order to decrease the multi-class surrogate risk down to its global minimum. On the one hand, using recent advances in boosting theory, we have proved that this algorithm converges fast to the global optimum of the risk function, thus providing an empirical upperbound to the training error. On the other hand, our training strategy exploits the boosting principle for explicitely selecting the most relevant prototypes among the training examples, thus removing confusing or irrelevant data that may impair classifier generalization and significantly reducing the computational cost of classification, which is logarithmic in the number of retained prototypes.

Experimental results have proved the effectiveness of our method compared to uniform *k*-NN voting, both in terms of classification precision, gaining up to 5% MAP on a challenging database of real-world images, and in terms of computational complexity, by reducing considerably the prototype set size. The precision improvement is significant considering that tests were carried out on small datasets (containing few thousands images), while boosting typically gets as better performances as the number

of weak hypotheses available gets larger. Furthermore, performances of MLNN are comparable to those of a state-of-the-art metric learning algorithm, *i.e.*, ITML, on UCI datasets, whereas our method is outperformed by support vector machines (SVM) on image categorization. Indeed, although our method overcomes some major drawbacks of k-NN, like uniform voting, uncertain classification bound and computational cost, it still suffers from a common limit of prototype methods when using high-dimensional feature vectors, *i.e.*, the well-known "curse of dimensionality" that induces "hubs" and makes k-NN distances "collapse", thus dramatically reducing the discriminative ability [RNI09].

PART IV

CLASSIFICATION OF MEDICAL IMAGES

- CHAPTER 6 -

CONTENT-BASED MEDICAL IMAGE CATEGORIZATION

Because of the increasing amount of medical image data available, locating relevant information in an efficient way is the most critical challenge when managing such huge collections. This problem involves analyzing the content of images, classifying them according to a pertaining set of labels, and searching for them effectively. Thus, recent research in image management has devoted much effort on developing tools and systems for content-based image analysis, indexing and searching. These tasks are inherently hard to solve, as they aim at using the low-level visual information in order to provide a meaningful semantic interpretation of the image content, *i.e.*, useful for computer-aided diagnosis systems. In particular, a crucial step involved in these systems is represented by the automatic annotation of images. For instance, when dealing with radiologic images, implementing automatic tools able to recognize the acquisition modality, the body orientation, the body region and the biological system examined can significantly help to improve the quality and efficiency of searching image collections for diagnostic purposes. Indeed, automatic annotation of medical images is expected to enable two major improvements over the traditional manual labeling. The first consists in speeding up the search in huge archives, where the amount of information available is consistently growing. The second is related to improving the quality of query results, *i.e.*, making the annotation process more reliable and consistent, by compensating errors induced in the tag assignment by manual classification.

In this chapter we concentrate on the task of automatic annotation of a radiologic database, which includes intra- and inter-individual variance and diseases. The main challenge when dealing with such data is represented by intra-class and inter-class variations that characterize these data. Indeed, images annotated with the same label may present significant visual differences, whereas images belonging to different classes may look very similar. On the one hand, higher visual intra-class variability im-

proves difficulty of more generic recognition tasks related to such images, like *body part recognition*, which we consider in the following sections. On the other hand, when defining more specific categories, *e.g.*, taking into account body orientation or biological system as well as body region, the classification task has to cope with the problem of inter-class variability, which is considerably reduced, making categories significantly overlap in the visual feature space.

In this chapter, we first present a brief survey of the recent literature on medical image classification (Sec. 6.1). Then, we depict our MLNN approach adapted to medical image categorization (Sec. 6.2) and describe a benchmark radiographic image dataset, IRMA, which has been widely used for validation purposes in the image retrieval and classification community (Sec. 6.3). Finally, we refer to our own work on medical image classification by reporting experimental results on a modified database of radiographs we extracted from the IRMA dataset, focusing on the task of body part recognition more specifically (Sec. 6.4).

§ 6.1 State-of-the-art medical image classification

Automatic categorization of medical images is a challenging task that can be of crucial importance when managing real data collections for the clinical routine. As pointed out by Tommasi and Deselaers [TD10], manually setting the tags of DICOM headers for medical radiographs may not be a reliable procedure, as some entries are either missing, false, or do not describe the anatomic region precisely. Moreover, visual information in images cannot be always represented by a textual description satisfactorily. Such issues have led to significant research interest in automatic methods for indexing medical images based on the analysis and description of their visual content, as attested by recent extensive evaluation challenges, like the ImageCLEF classification competition [TD10].

Although state-of-the-art approaches are now able to provide reliable tools for – say – recognizing body parts in radiographs automatically, most existing techniques are still too computationally expensive when dealing with large databases containing many classes, or they require many training images in order to reach satisfactory classification performances. Moreover, extracting appropriate visual descriptors for a specific task is still an open research challenge, which consists in reducing the gap between the semantic interpretation of the visual content and the low-level representation of images [SWS+00, MMBG04].

Many successful techniques for automatic categorization of medical images rely on the same kind of descriptors as some popular methods for

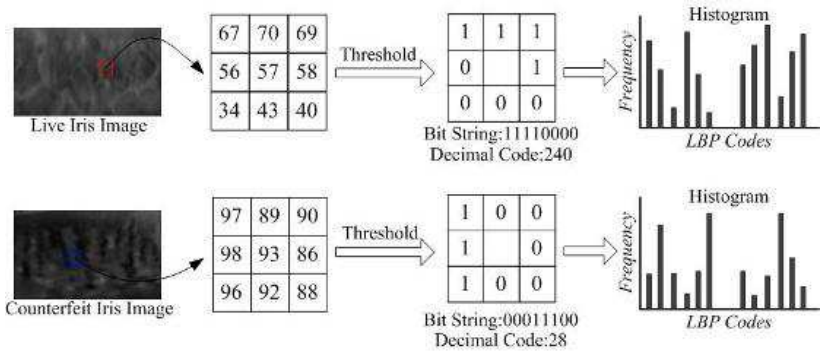


Figure 6.1: *Extraction of Local Binary Patterns (LBP) descriptors from images [OPM02].*

generic image classification. In particular, the Bag-of-Features (BoF) scheme based on local descriptors (like SIFT) has shown good performances for representing relevant information of medical images for the automatic classification purpose. *E.g.*, recently, Avni et al [AGS+10] have successfully applied the BoF approach to the classification of X-Ray images. Namely, their method uses histograms of vector quantized SIFT descriptors for automatic organ recognition. Furthermore, they also apply their method to discriminate between healthy and pathologic cases on a set of chest radiographs. SIFT descriptors and the BoF representation have been also successfully applied to a binary classification problem related to endomicroscopic images, thus enabling to discriminate between neoplastic and benign cases with high accuracy [AVP+09].

Another tool for feature extraction in medical image retrieval and classification is the Local Binary Patterns (LBP) descriptor [OPM02]. This descriptor is based on a very simple idea to efficiently describe the local texture pattern around a pixel. Indeed, LBP consists in a binary code that is obtained by thresholding a neighborhood by the gray value of its center, as displayed in Fig. 6.1. The effectiveness of LBP for radiographic images has been recently demonstrated by Jeanne et al [JUJ09], who have found significant performance improvement over other common visual descriptors. This is mainly due to the luminance invariance of LBP, which considerably helps to correctly classify image categories with significant exposure changes. Several modifications of LBP have been also proposed in order to further improve classification accuracy on X-Ray images. For instance, Unay et al [USE+09] have proposed PCA as a valuable feature selection method for reducing the computational complexity of supervised learning relying on such texture descriptors.

Finally, the best performing methods for medical image classification are those relying on multi-cue fusion, that is the combination of different types of descriptors, *e.g.*, local and global features. Generally, this strategy

allows to capture most of the information that is relevant to classify images according to their content. For instance, Tommasi et al [TOC08a] have proposed an effective method for integrating two different local cues that describe structural and textural information of image patches. The combination of these two descriptors is carried out by either concatenating them into a single vector or combining the output of two kernel machines separately trained on them. Tao et al [TPJ+09] have successfully addressed the problem of automatically recognizing the projection view of chest radiographs (posteroanterior/anteroposterior (PA-AP) and lateral (LAT) views), by using a method that is based on sparse aggregation of learned local appearance cues.

Independently on the kind of image descriptors used, most state-of-the-art approaches to medical image classification rely on discriminative learning tools like support vector machines (SVM) [DD09]. However, other learning approaches have been also proposed in order to deal with specific problems. For instance, Yang et al [YJM+10] have recently proposed to define the medical image categorization problem in a boosting framework, where a distance metric is learned in order to enforce the visual similarity between images that are semantically related to each others.

In this chapter, we propose a new approach to radiographic image classification, which also relies on a boosting framework, while not involving to learn any distance metric. In particular, we adapted the MLNN learning algorithm described in Chap. 5 to the classification of a standard database of X-Ray images. In the following sections, we first describe this dataset, then we discuss experimental results of our MLNN method and comparison with the baseline *k*-NN and state-of-the-art SVM classification.

§ 6.2 METHOD

In this thesis, we have proposed a novel supervised learning algorithm, MLNN, which brings the classic *prototype-based* classification rules, like *k*-NN, into a multi-class boosting framework. The main contribution of our method is to "boost" the accuracy of image classification by simply learning the "prototypical importance" of annotated examples, while retaining a sparse subset of these data as prototypes for the classification phase. Such an importance learning strategy allows us to reduce the computational cost of classification significantly. Moreover, compared to most state-of-the-art discriminative learning techniques, which rely on splitting a multi-class problem into multiple binary problems, the computational cost of MLNN is significantly lower, thus generally enabling a better accuracy/time trade-off.

We tested our method for the task of automatic classification of medical X-Ray images, which gives rise to some critical issues, such as low *inter-*

class variability, high *intra-class variability* and large *class imbalance*, which may severely impact on classification performances [TOC08b]. In order to obtain a robust and meaningful representation of the medical image content, we used the Bag-of-Features scheme combined with dense SIFT descriptors extracted at multiple resolution levels (similarly to what proposed in Chap. 5). While such local descriptors enable a fine representation of body tissue details, the Bag-of-Features aggregation strategy provides a compact descriptor of the overall image, thus allowing one to use discriminative learning algorithms that need one-to-one correspondence between labeled examples and feature vectors.

A crucial problem when dealing with the Bag-of-Features histogram descriptors is to define a similarity measure being appropriate for prototype-based classification. In this work, we propose to use the Histogram Intersection kernel [SB91], which has been shown to provide better classification performances than classic Euclidean distance when dealing with histogram-based descriptors. In particular, using the notation of Sec. 5.2.4, the Histogram Intersection kernel is defined as follows:

$$K_{\mathrm{HI}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1 - \frac{1}{2} \sum_{h=1}^{d} \frac{|x_{ih} - x_{jh}|}{x_{ih} + x_{jh}} \ , \tag{6.1}$$

where d is the number of features and both descriptors $\boldsymbol{x}_i, \boldsymbol{x}_j$ are ℓ_1-normalized histograms. Thanks to this latter condition, kernel (6.1) is closely related to the ℓ_1 distance between the two descriptors. In particular, the following linear relationship holds:

$$K_{\mathrm{HI}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1 - \frac{1}{2} \, ||\boldsymbol{x}_i - \boldsymbol{x}_j||_1 \ . \tag{6.2}$$

Furthermore, in order to reduce the computational cost significantly, we propose to "truncate" kernel (6.1) to the k-NN, similarly as for Gaussian kernel in Sec. 5.2.4. Notice that, according to (6.2), the k-nearest neighborhood relationship is to be defined in the sense of the ℓ_1 distance, so as to ensure that k-NN are those descriptors that maximize the Histogram Intersection similarity measure. Therefore, denoting as $\rho_k^1(\boldsymbol{x})$ the ℓ_1 distance of the k-th nearest neighbor to $\boldsymbol{x}$, we define the following *truncated Histogram Intersection* (tHI) kernel:

$$K_{\mathrm{tHI}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = K_{\mathrm{HI}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \cdot \left[\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||_1}{\rho_k^1(\boldsymbol{x}_i)} \leq 1 \right] \ , \tag{6.3}$$

where the square brackets denote the indicator function that selects the k-NN. In practice, we only need to search for the k-NN of $\boldsymbol{x}$ in the sense of ℓ_1 norm, thus avoiding to compute the entire kernel matrix (6.1). This makes the computation much more efficient, as k-NN search can be performed

using fast data structures such as *kd*-trees, which are able to deal with any ℓ_p metric distance [AMN+98]. In this case, kernel (6.3) can be viewed as a simple weighting factor that takes into account the real-valued similarity between an example and its k-NN.

§ 6.3 DATABASE AND SETTINGS

We focused on the automatic classification of radiographic images, which has been an open challenge in the computer vision community in the recent years. In particular, the task of automatic annotation of radiographs has been part of the well known ImageCLEF competition, which is a broad evaluation campaign that has been carried out for several years with the aim of benchmarking advances in the area of visual data management. The ImageCLEF medical image annotation task focuses on the automatic classification of X-Ray images, which were collected from the medical routine. The last edition of this benchmark database (that of 2009) contains more than 10,000 images. In order to annotate these images in a relevant way for real medical applications, a complex hierarchical labeling scheme, called IRMA (Image Retrieval in Medical Applications) code [LSK+03], was proposed. Namely, the IRMA annotation scheme consists of four parts that encode *imaging modality*, *body orientation*, *anatomical region* and *biological system* examined. Each part of the code is then organized in a mono-hierarchical way, such that ambiguities in textual classification are avoided. In particular, in the last editions of the benchmark (2008 and 2009), a special scoring scheme was defined for comparing performances of the different methods, such as to really exploit this labeling hierarchy, by penalizing wrong classification in the highest hierarchy positions over the lowest ones, as well as penalizing the false category associations over the assignment of unknown codes [DD09].

The data we consider in the following come from the ImageCLEF 2009 database for the medical annotation task. This database contains 12,677 X-Ray images, which were manually annotated according to the IRMA code, such that 193 distinct classes are to be considered when taking into account the four annotation codes. Image categories in the IRMA database are very unbalanced, *i.e.*, they contain a largely variable number of images, thus making the full annotation task very challenging.

In our experiments, we focused on a reduced dataset, which was specifically tailored to the task of *body part recognition*. Indeed, we removed images that were not related to a specific organ or body part and took into account only the annotation codes related to the main *anatomical regions*, thus drastically reducing the number of categories. The new dataset we obtained is called IRMA-12, and contains 12 categories of radiographs whose labels refer to the anatomical regions examined. In particular, we retained

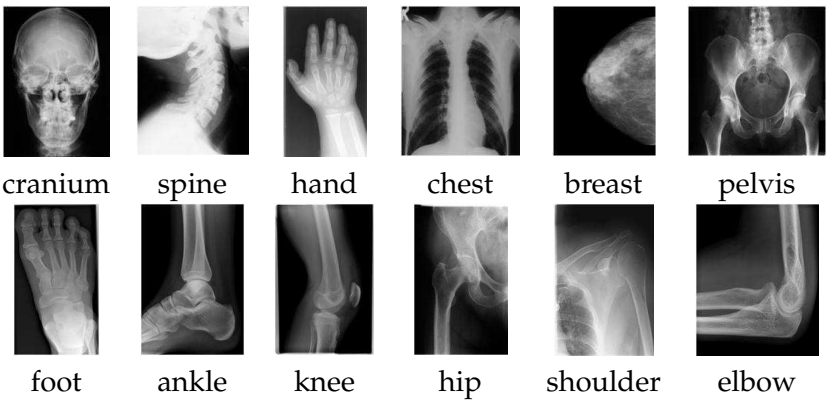


Figure 6.2: *The 12 categories of radiographic images from IRMA database [ima] we considered in our experiments.*

8981 images from the original dataset (more than 70%). These images were grouped into the following categories: *Ankle and lower leg* (565 images), *Breast* (332 images), *Chest* (3500 images), *Cranium* (1129 images), *Elbow* (259 images), *Foot* (480 images), *Hand* (811 images), *Hip and upper leg* (259 images), *Knee and middle leg* (575 images), *Pelvis* (277 images), *Shoulder and upper arm* (355 images), *Spine* (439 images). Exemplar images of these twelve classes are displayed in Fig. 6.2. Notice that our IRMA-12 database is highly unbalanced, as the number of images per category ranges from 259 up to 3500.

In order to test the effectiveness of our method we removed the unbalancing between classes, that may affect the evaluation, and we reserve to investigate the effects of class unbalancing separately. Namely, we retained 250 random images per category, thus forming a balanced database of 3,000 images, *i.e.*, uniformly distributed over the classes. We used this database for carrying out 10 fold-cross validation experiments. Thus, we split the dataset in ten subsets of equal size, *i.e.*, each one containing 300 images, and run ten classification experiments. Each time, one different fold was used as testing set, after learning on the remaining nine folds. The results reported in the following section refer to averaging the classification performances over these ten folds.

§ 6.4 Experiments

In this section, we report our experimental validation of MLNN on the database of medical radiographs described in Sec. 6.3. Indeed, we focused on the task of body part recognition, which is expected to represent a crucial preliminary step in a real content-based *computer-aided diagnosis* sys-

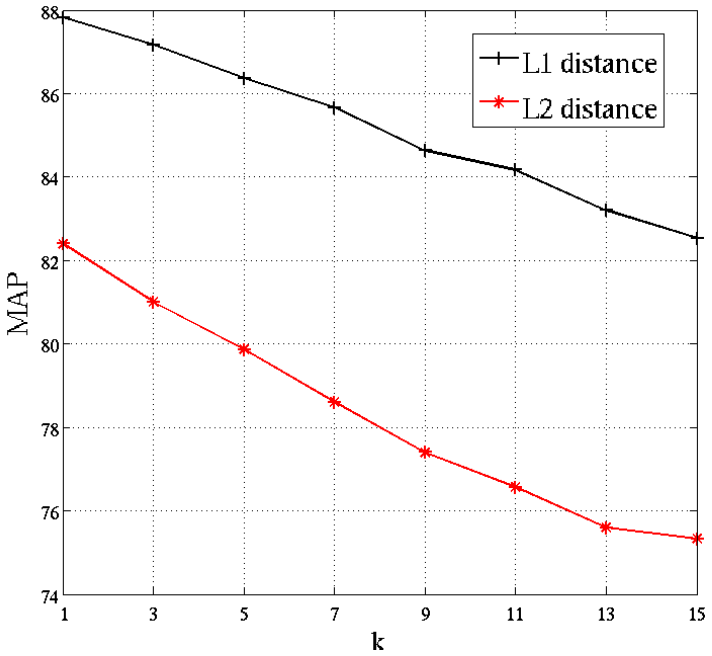


Figure 6.3: *Comparison of k-NN classification using ℓ_1 and ℓ_2 distance as a function of k.*

tem. In particular, we tested MLNN with the implementation described in Sec. 6.2, and compared our results with those of both classic k-NN classification, in order to quantify the improvement provided by our method, and support vector machines, which provide the best state-of-the-art performances up to date.

First of all, we investigated the effect of changing the distance metric for prototype-based classification, replacing the classic Euclidean distance by the ℓ_1 distance, as proposed in Sec. 6.2. Indeed, due to relationship (6.2), this is equivalent to computing the Histogram Intersection similarity measure on ℓ_1-normalized Bag-of-Features descriptors and then finding the k largest scores. In Fig. 6.3 we report cross-validation performances (in terms of MAP) of classic k-NN classification when varying the value of k. In particular, we compare ℓ_1 and ℓ_2 distances as dissimilarity criteria for k-NN search. Besides showing a linear performance decrease when increasing k, those results show the significant improvement in classification accuracy (more than 6%) provided by using the ℓ_1 distance, thus confirming the Histogram Intersection kernel as a valuable similarity criterion for Bag-of-Feature–based classification.

Then, we evaluated performances of our MLNN algorithm using the implementation detailed in Sec. 6.2. In particular, we carried out experiments of MLNN using the Histogram Intersection kernel truncated at k-

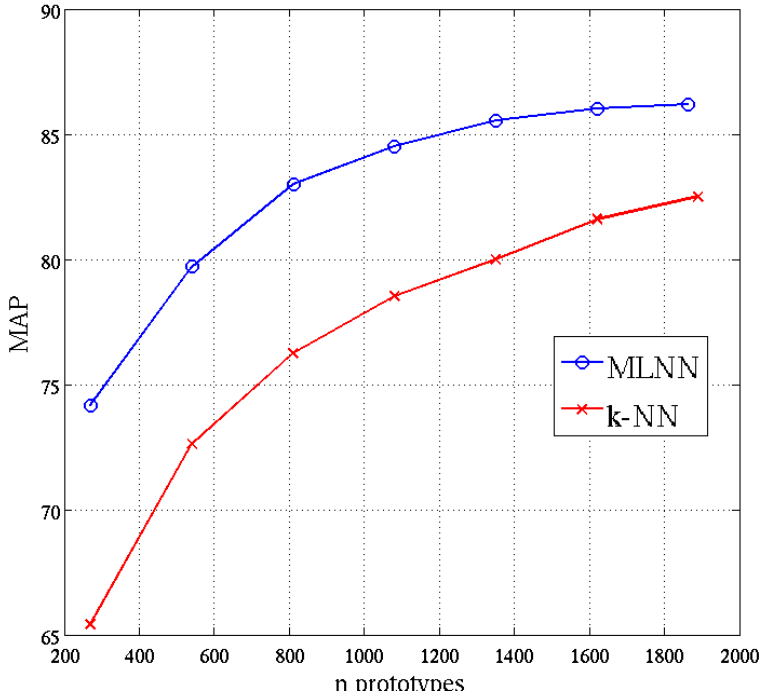


Figure 6.4: *Performances of* MLNN *on the IRMA database.*

NN, when varying the proportion of retained prototypes. Cross-validation results are displayed in Fig. 6.4, where MLNN is compared with k-NN classification (in both cases $k = 10$). (We implemented data reduction for k-NN as a random selection of a fixed proportion of prototypes, and in each experiment we averaged results over 10 different random subsamples.) Such plots can be viewed as the cost-accuracy trade-off for those prototype-based classification methods, since their computational complexity only depends on the number of prototypes (for k fixed). In particular, both MLNN and k-NN methods have computational cost of classification logarithmically increasing with the number of prototypes. Comparison with k-NN results shows that our method yields better performances for a fixed number of prototypes (up to 9% gap), while providing comparable performances using much fewer prototypes (86% MAP for 50% of prototypes), thus confirming the ability of MLNN to learn the most relevant prototypes to use at classification time. This data reduction ability allows us to reduce the computational cost of classification significantly, while keeping satisfactory performances compared to those of full k-NN classification (*i.e.*, k-NN using all the training data as prototypes).

In Tab. 6.1 we show the confusion table for MLNN using the Histogram Intersection Kernel truncated at k-NN (with k=10). Cross-validation results are reported in terms of recognition rate (percentage) per category. Notice

Table 6.1: *An example confusion table for the radiographic image dataset. Results correspond to 10-fold cross-validation using* MLNN *with the Histogram Intersection Kernel truncated at* $k = 10$*. Legend of categories:* A *Ankle and lower leg,* B *Breast,* Ch *Chest,* Cr *Cranium,* E *Elbow,* F *Foot,* Ha *Hand,* Hi *Hip and upper leg,* K *Knee and middle leg,* P *Pelvis,* Sh *Shoulder and upper arm,* Sp *Spine.*

	A	B	Ch	Cr	E	F	Ha	Hi	K	P	Sh	Sp
A	**86.4**	1.2	0.0	0.8	2.4	2.0	2.0	1.6	2.0	1.6	0.0	0.0
B	0.0	**96.8**	0.0	0.4	0.0	0.0	0.0	0.4	0.0	0.4	0.8	1.2
Ch	0.0	0.0	**98.4**	1.2	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0
Cr	0.0	0.0	0.4	**96.4**	0.0	0.0	0.0	0.0	0.0	2.8	0.0	0.4
E	4.8	0.8	0.0	2.8	**67.6**	2.4	4.0	4.0	7.2	4.8	1.2	0.4
F	0.8	0.8	0.0	1.2	1.6	**86.8**	5.2	1.2	0.0	1.2	1.2	0.0
Ha	0.0	0.0	0.0	1.2	0.8	7.6	**90.0**	0.0	0.0	0.0	0.0	0.4
Hi	1.2	1.2	0	0.8	2.4	0.0	2.8	**66.8**	2.8	19.6	0.4	2.0
K	7.2	0.0	0.0	0.0	4.0	0.8	0.8	4.4	**82.8**	0.0	0.0	0.0
P	0.0	0.4	0.4	0.4	0.0	0.0	0.0	3.2	0.0	**95.6**	0.0	0.0
Sh	0.8	1.2	1.2	2.8	1.2	0.8	1.2	3.6	0.0	0.8	**84.4**	2.0
Sp	0.0	0.0	4.4	6.4	0.0	0.0	0.0	0.4	0.0	8.0	0.4	**80.4**

the high confusion between category *Hip and upper leg* and *Pelvis* (19.6% of images of the former class are misclassified into the second). This can be easily understood by considering the strong visual similarity between prototypical images of the two categories, which exhibit low inter-class variability, as shown in Fig. 6.5.

Finally, we compared MLNN performances to those of support vector machines (SVM). SVM classification is the most widely used state-of-the-art learning tool for medical image classification, as it generally provides the best performances [DD09]. Nevertheless, SVM are still affected by two major drawbacks that severely impact on the time efficiency of both training and classification when dealing with large image collections, such as those tipically involved in medical information systems. The first one is represented by the high computational cost of both training and test when using non-linear kernels, such as the common Gaussian kernel, whereas the second consists in the need for splitting multi-class learning problems into multiple binary problems to treat distinctly, thus multiplying the computation time by the number of classes. Namely, the computational complexity of non-linear SVM classification with n support vectors (prototypes) in dimension d and C classes is $\mathcal{O}(Cnd)$, that is linear in the size of prototype dataset and in the number of classes. Our MLNN method with k-NN-truncated kernel, instead, has computational complexity only depending on k-NN search, that is $\mathcal{O}(d \log n)$ when using appropriate data structures like kD-trees for fast search [AMN+98]. A common solution adopted by SVM-based approaches for speeding up classification is to use linear SVM, which allows for much faster training and classification. In particular, the computational complexity of linear SVM classification is $\mathcal{O}(Cd)$, *i.e.*, independent on the number of support vectors, while still being linear in the

(a) (b)

Figure 6.5: *Two exemplar images from categories (a)* hip and upper leg, *and (b)* pelvis. *These two categories feature low inter-class variability, due to the presence of the hip joint in prototypical images of both classes. This justifies the high confusion rate between the two classes.*

Table 6.2: *Performance comparison of* MLNN *with standard* k-NN *and SVM classification on IRMA images (cross-validation).*

method	mAP	C	d	n	complexity
MLNN	84.52	12	500	1080	$\mathcal{O}(d \log n)$
MLNN	86.20	12	500	1863	$\mathcal{O}(d \log n)$
k-NN	84.37	12	500	2700	$\mathcal{O}(d \log n)$
kernel SVM	92.70	12	500	248	$\mathcal{O}(Cnd)$
linear SVM	88.63	12	500	315	$\mathcal{O}(Cd)$

number of classes. However, this computational speed-up is generally obtained at the price of worse classification performances. In Tab. 6.2, we report cross-validation performances (in terms of MAP) of our method, k-NN and SVM (both Gaussian kernel–based and linear), along with their computational complexity and the size of the prototype dataset (*i.e.*, number of support vectors for SVM). Although kernel-based SVM still yield the best performances, their computational cost is significantly higher than that of prototype-based classification, due to the linear dependence both on the number of support vectors and on the number of classes. Indeed, even though MLNN is outperformed by SVM, it allows for much faster classification, due to the logarithmic dependence of the computational complexity on the number of prototypes, which can be drastically reduced without impairing the classification precision significantly and still outperforming the baseline k-NN. (See the first two lines in Tab. 6.2.)

§ 6.5 CONCLUSION

In this chapter, we have addressed the task of medical image categorization, particularly focusing on recognition of body parts in radiographic images. Namely, we extracted a balanced database containing 12 body part categories from a benchmark database that has been widely used in the recent years. We carried out experiments with the purpose of validating our MLNN method on such a challenging task. In particular, we tested our algorithm using the kernel-based implementation described in Sec. 6.2, and compared our results with those of both baseline k-NN and SVM classification. On the one hand, experimental results enlight the crucial role of the distance metric when comparing histogram-based descriptors like the state-of-the-art Bag-of-Features descriptors, suggesting that intersection distance measures like the ℓ_1 distance are the most appropriate ones. On the other hand, results show that MLNN method, although being still outperformed by kernel-based SVM significantly, is able to better deal with the precision-speed trade-off, enabling for much faster classification than kernel-based machines while yielding satisfactory performances. This makes MLNN a good candidate algorithm for addressing the medical image classification task when facing with hard computation time constraints.

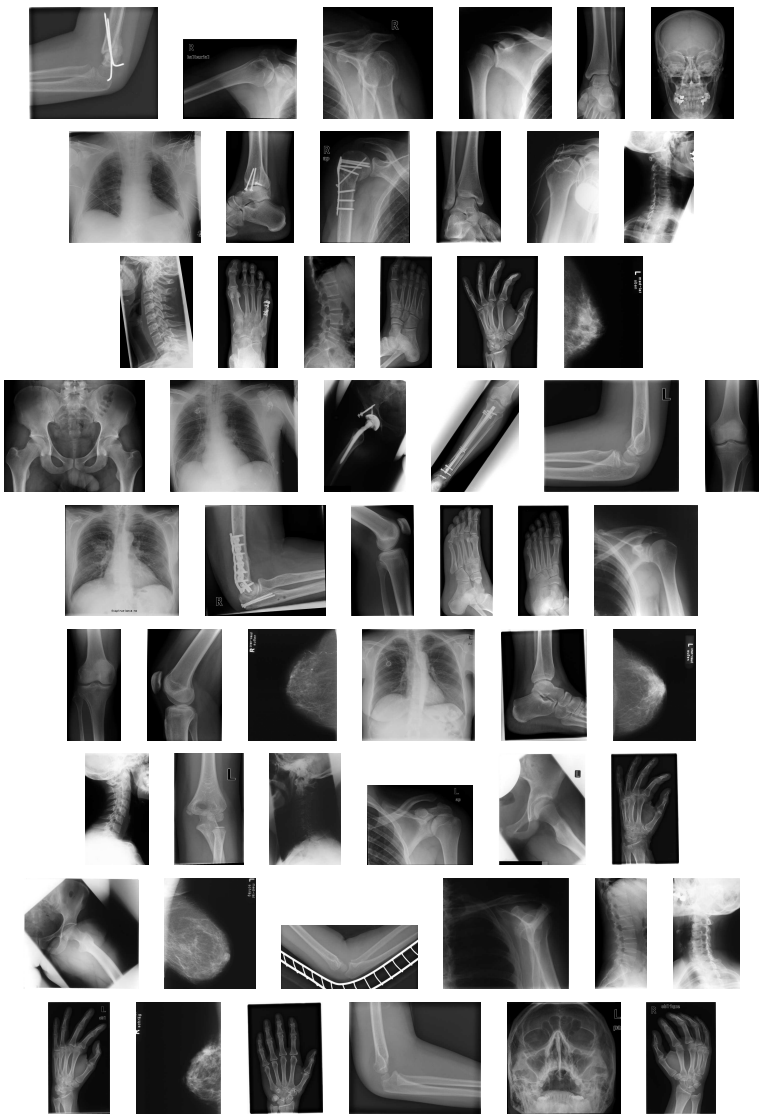


Figure 6.6: *Some examples of prototypes selected during the training phase (notice the variability in the number of prototypes per category).*

PART V

GENERAL CONCLUSION

- CHAPTER 7 -

CONCLUSION

In this thesis, we have focused on some major tasks involved in automatic processing systems for *indexing*, *retrieval* and *classification* of image collections. These tasks range from the extraction of meaningful *descriptors* of the visual content, *i.e.*, able to represent visual information in a relevant way, through the efficient organization of such image representations in *data structures* adapted to fast similarity-based retrieval, to *supervised learning* of classification rules based on similarity to relevant prototypes of image categories. Indeed, in the first part of this thesis, we have first proposed a new scalable descriptor (SMP) for image indexing that is related to an information-theoretic similarity measure, then we have described our generalization of two well-known data structures (BB-trees and Bvp-trees) for fast similarity retrieval when using Bregman divergences as distorsion measures in the feature space. However, the "core" of this thesis has been devoted to the third mentioned point, *i.e.*, image classification, because defining tools for tackling this task in an efficient and reliable way is a crucial challenge in nowadays applications, that require to manage huge collections of images effectively (*e.g.*, medical image systems and multimedia repositories).

For this purpose, we have proposed a *novel* learning algorithm (UNN) that is methodologically inspired by the *prototypical classification* principle, which has been proposed as a reasonable model of the way human themselves accomplish the visual categorization task [Ros73]. According to this principle, deciding the label of an unknown image consists in retrieving the most similar instances from a learned (possibly sparse) set of prototypes. Thus we started from the classic k-NN voting rule, which implements this idea in the simplest way, and generalized it in a *boosting* framework, which allows us to "leverage" votes from the closest prototypes as weak classifiers, while constraining the training error under certain convenient bounds that enable significant improvements in classification performances. Be-

sides weighting contributions from the k-nearest neighbors according to a "soft" voting rule, our UNN algorithm is also able to learn a sparse subset of the annotated data according to their "prototypical relevance", thus providing better resilience to annotation ambiguities. The basis idea is to rely the relevance of an example to the local "coherence" of this reciprocal neighborhood, *i.e.*, to the probability of giving uncorrect vote to its reciprocal k-NN. In this sense, our method shares a common philosophical principle with the well-known Google PageRank algorithm for document ranking [BP98], although this latter is based on a completely different optimization problem. Indeed, UNN exploits the ground-truth annotation of the training images in order to estimate their "popularity" for a given category, *i.e.*, the local density of neighbor examples that "agree" with their own category.

Moreover, such a prototype selection, which can be tuned at training time directly, by setting the number of boosting iterations, impacts on computational efficiency very favourably, due to reducing the k-NN search space considerably.

Then, we have further extended our approach to generic kernel density classification, in order to improve the precision of local class density estimation that underlies our prototype-based classification rule. In particular, we have proposed a novel algorithm, MLNN, as a framework for leveraging such classifiers in an inherently multi-class fashion, which avoids the common trick needed for most discriminative learning methods, *i.e.*, splitting the multi-class problem into multiple binary problems. This enables significant speed-up in both training and classification over – say – one-versus-all strategies, whose computational cost is linear in the number of categories. Such a speed-up is obtained while maintaining satisfactory performances, *e.g.*, yielding precision equal to that of classic k-NN using as few as 10% or 20% of prototypes, or outperforming k-NN by 5% to 10% (in terms of MAP) using up to 50% of prototypes.

Globally, experimental results suggest that our novel learning framework for boosting k-NN and prototype-based classification is very promising for applications requiring a suitable trade-off between precision and computational speed. In particular, our multiclass algorithm MLNN meets the requirement of scalability in the database size and in the number of categories, which has become crucial in nowadays applications involving huge image collections and real-world annotation tasks. Furthermore, our MLNN framework has the advantage of operating "on the top" of both visual content representation and similarity/dissimilarity measures used for learning the voting rule, thus being straightforwardly adapted to more effective image representation schemes, like visual vocabularies with soft assignement [PCI+08], aggregation of local descriptors using spatial pyramids [LSP06], Fisher kernels [PSM10] or VLAD [JDSP10], as well as metric learning techniques like ITML [DKJ+07].

Nevertheless, in terms of absolute performances, our method is still outperformed by state-of-the-art support vector machines (SVM) for the image classification task. Indeed, MLNN for image categorization is still affected by a major drawback of instance-based methods working in a high-dimensional feature space, that is the *curse of dimensionality*, which induces the emergence of "hubs" and collapsing distances between feature points, thus reducing the discriminant ability of k-NN and related voting methods. At the same time, our evaluation of most existing dimensionality reduction techniques, like PCA or pLSA, did not provide significant improvements, thus suggesting that this problem should be specifically redefined for high-dimensional descriptors arising from images, such as Gist or Bag-of-Features histograms, *e.g.*, investigating suitable manifold learning techniques.

However, better performances of SVM are obtained at a much higher computational cost, which makes this method very impractical on large databases containing many categories. This is mainly due to the high computational complexity of non linear SVM learning and classification, besides their binary nature, which obliges one to treat each class separately. Furthermore, classification performances of SVM are largely sensitive to value of the optimization parameter, that has to be selected accurately, *e.g.*, learning it via time-consuming cross-validation.

To summarize, our MLNN approach should be viewed as a general approach for improving k-NN and prototype-based classification methods by embbedding them in a boosting framework. This framework not only constrains those classifiers under certain classification bounds, thus generally improving testing performances significantly, but also reduces the computational cost at classification time, thus making MLNN a good candidate for replacing traditional voting rules in applications where fast computation is a critical requirement.

PART VI

APPENDICES

- APPENDIX A -

UNN APPENDIX

§ A.1 GENERIC UNN ALGORITHM

The general version of UNN is shown in Alg. 5 as a pseudocode. This algorithm induces the leveraged k-NN rule (4.10) for the broad class of surrogate losses meeting conditions of [BJM06], thus generalizing Alg. 3. Namely, loss function ψ is contrained to meet the following conditions:

(i) $\mathrm{im}(\psi) = \mathbb{R}_+$,

(ii) $\nabla_\psi(0) < 0$ (∇_ψ is the conventional derivative of ψ),

(iii) ψ is strictly convex and differentiable.

The two first conditions **(i)** and **(ii)** imply that ψ is *classification-calibrated*, *i.e.*, its local minimization is roughly tied up to that of the empirical risk [BJM06]. Furthermore, condition **(iii)** implies convenient algorithmic properties for the minimization of the surrogate risk [NN09c]. Three common examples of such *strictly convex losses* (SCS) have been shown in Eq. (4.6 – 4.8).

The main bottleneck of UNN is step **[I.1]**, as Eq. (A.2) is non-linear. However, this equation *always* admits a solution, finite under mild assumptions [NN09c]. In particular, in our case, δ_j is guaranteed to be finite when there is no total matching nor mismatching of example j's memberships with its reciprocal neighbors', for the class at hand. The second column of Table A.1 contains the solutions to Eq. (A.2) for the surrogate losses mentioned in Sec. 4.2.2. Those solutions are always exact for the exponential loss (ψ^{exp}) and squared loss (ψ^{squ}); for the logistic loss (ψ^{log}) that is exact when the weights in the the reciprocal neighborhood of j are the same, otherwise it is approximated. Since the starting weights are all the same,

Algorithm 5 Algorithm UNIVERSAL NEAREST NEIGHBORS UNN($\mathcal{S}, \psi$)

Input: $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m,\ \boldsymbol{x}_i \in \mathcal{X},\ \boldsymbol{y}_i \in \{-\frac{1}{C-1}, 1\}^C\}$, ψ meeting **(i)**, **(ii)**, **(iii)** (Sec. A.1);

Let $\mathrm{r}_{ij}^{(c)} \doteq \begin{cases} y_{ic} y_{jc} & \text{if} \quad \boldsymbol{x}_j \in \mathrm{NN}_k(\boldsymbol{x}_i) \\ 0 & \text{otherwise} \end{cases}$, $\forall i, j = 1, 2, ..., m,\ c = 1, 2, ..., C$;

for $c = 1, 2, ..., C$ **do**

Let $\alpha_{jc} \leftarrow 0, \quad \forall j = 1, 2, ..., m$

Let $w_i \leftarrow -\nabla_\psi(0) \in \mathrm{R}_{+*}^m, \quad \forall i = 1, 2, ..., m$

for $t = 1, 2, ..., T$ **do**

[I.0] Let $j \leftarrow \mathrm{WIC}(\{1, 2, ..., m\}, t)$

[I.1] Let

$$w_j^+ = \sum_{i:\mathrm{r}_{ij}^{(c)} > 0} w_i,\ w_j^- = \sum_{i:\mathrm{r}_{ij}^{(c)} < 0} w_i\ , \tag{A.1}$$

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

$$\sum_{i=1}^m \mathrm{r}_{ij}^{(c)} \nabla_\psi \left(\delta_j \mathrm{r}_{ij}^{(c)} + \nabla_\psi^{-1}(-w_i) \right) = 0\ ; \tag{A.2}$$

[I.2] $\forall i\ :\ \boldsymbol{x}_j \in \mathrm{NN}_k(\boldsymbol{x}_i)$, let

$$w_i \leftarrow -\nabla_\psi \left(\delta_j \mathrm{r}_{ij}^{(c)} + \nabla_\psi^{-1}(-w_i) \right)\ . \tag{A.3}$$

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

Output: $h_c(\boldsymbol{x}) = \sum_{t=1}^T \alpha_{tc} y_{tc} \left[\boldsymbol{x}_t \in \mathrm{NN}_k(\boldsymbol{x}) \right], \quad \forall c = 1, 2, ..., C$

exactness can be guaranteed during a large number of inner rounds depending on which order is used to choice the examples. Table A.1 helps to formalize the finiteness condition on δ_j mentioned above: when either sum of weights in (A.1) is zero, the solutions in the first and third line of Table A.1 are not finite. A simple strategy to cope with numerical problems arising from such situations is that proposed by [SS99]. (See Sec. 4.2.4.) Table A.1 also shows how the weight update rule (B.4) specializes for the mentioned losses.

Proofsketch of Theorem 3 We show that UNN converges to the global optimum of any surrogate risk (Sec. 4.2.5). For this purpose, let us consider the surrogate risk (5.13) for a given class $c = 1, 2, ..., C$:

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

In this section, we use the following notations:

Table A.1: *Three common loss functions and the corresponding solutions δ_j of (A.2) and w_i of (B.4). (Vector $\boldsymbol{r}_j^{(c)}$ designates column j of $\mathrm{R}^{(c)}$ and $||.||_1$ denotes 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	δ_j in (A.2)	w_i in (B.4)	Opt
$\psi^{\exp} \doteq \exp(-x)$	$\frac{1}{2}\log\left(\frac{w_j^{(c)+}}{w_j^{(c)-}}\right)$	$w_i \exp\left(-\delta_j \mathrm{r}_{ij}^{(c)}\right)$	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
$\psi^{\log} \doteq \log(1+\exp(-x))$	$\log\left(\frac{w_j^{(c)+}}{w_j^{(c)-}}\right)$	$\frac{w_i \exp\left(-\delta_j \mathrm{r}_{ij}^{(c)}\right)}{1 - w_i\left(1+\exp\left(-\delta_j \mathrm{r}_{ij}^{(c)}\right)\right)}$	S

- $\tilde{\psi}(x) \doteq \psi^\star(-x)$, where $\psi^\star(x) \doteq x\nabla_\psi^{-1}(x) - \psi(\nabla_\psi^{-1}(x))$ is the Legendre conjugate of ψ, which is strictly convex and differentiable as well. ($\tilde{\psi}$ is related to ψ 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 $\tilde{\psi}$ [NN09c].

Let $\boldsymbol{w}_t$ denote the t^{th} weight vector inside the "**for** c" loop of Alg. 5 (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 [NN09c]:

$$\psi(\varrho(\boldsymbol{h}_t^\ell, i, c)) \quad = \quad g + D_{\tilde{\psi}}\left(0||w_{ti}\right) \ , \tag{A.5}$$

where $g(m) \doteq -\tilde{\psi}(0)$ does not depend on the k-NN rule. In particular, Eq. (A.5) makes the connection between the real-valued classification problem and a geometric problem in the non-metric space of weights. Moreover, Eq. (A.5) 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. (A.5) in Eq. (A.4), and computing δ_j in Eq. (A.2) so as to bring $\boldsymbol{h}_{t+1}^\ell$ from $\boldsymbol{h}_t^\ell$, we obtain the following identity:

$$\varepsilon_c^\psi(\boldsymbol{h}_{t+1}^\ell, \mathcal{S}) - \varepsilon_c^\psi(\boldsymbol{h}_t^\ell, \mathcal{S}) = -\frac{1}{m}\sum_{i=1}^{m} D_{\tilde{\psi}}\left(w_{(t+1)i}||w_{ti}\right). \tag{A.6}$$

Since Bregman divergences are non negative and meet the identity of the indiscernibles, (A.6) implies that steps **[I.1]** — **[I.3]** *guarantee* the decrease of (A.4) as long as $\delta_j \neq 0$. But (A.4) is lowerbounded, hence UNN must converge.

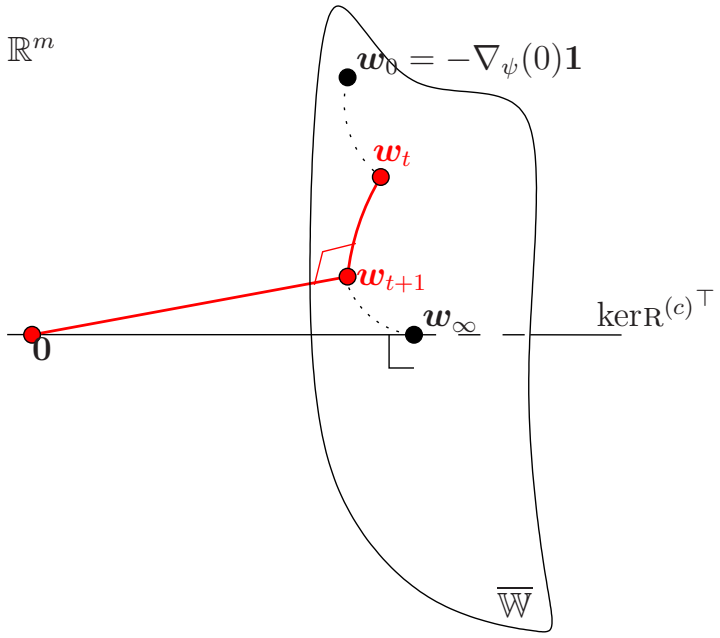


Figure A.1: *A geometric view of how* UNN *converges to the global optimum of (5.13). (See Appendix for details and notations.)*

In addition, it converges to the global optimum of the risk (5.13). Since predictions for each class are independent, the proof consists in showing that (A.4) 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 [NN09c], (A.2) and (B.4) 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 $\mathbb{W}$ (*i.e.*, the manifold where $\boldsymbol{w}$'s live), are provably Bregman orthogonal [NN09c], thus yielding:

$$\underbrace{\sum_{i=1}^m D_{\tilde{\psi}}\left(0||w_i\right)}_{m\varepsilon_c^\psi(\boldsymbol{h}^\ell,\mathcal{S})-mg} = \underbrace{\sum_{i=1}^m D_{\tilde{\psi}}\left(0||w_{\infty i}\right)}_{m\varepsilon_c^\psi(\boldsymbol{h}^\ell_\infty,\mathcal{S})-mg} + \underbrace{\sum_{i=1}^m D_{\tilde{\psi}}\left(w_{\infty i}||w_i\right)}_{\geq 0}, \forall \boldsymbol{w} \in \overline{\mathbb{W}} \ . \tag{A.7}$$

Underbraces use (A.5) in (A.4), and $\boldsymbol{h}^\ell$ is a leveraged k-NN rule corresponding to $\boldsymbol{w}$. One obtains that $\boldsymbol{h}^\ell_\infty$ achieves the global minimum of (A.4), as claimed.

The proofsketch is graphically summarized in Figure A.1. In particular, two crucial *Bregman orthogonalities* are mentioned [NN09c]. The red one symbolizes:

$$\begin{aligned}\sum_{i=1}^{m} D_{\tilde{\psi}}\left(0||w_{ti}\right) &= \sum_{i=1}^{m} D_{\tilde{\psi}}\left(0||w_{(t+1)i}\right) \\ &\quad + \sum_{i=1}^{m} D_{\tilde{\psi}}\left(w_{(t+1)i}||w_{ti}\right) ,\end{aligned} \tag{A.8}$$

which is equivalent to (A.6). The black one on $\boldsymbol{w}_\infty$ is (A.7).

Proofsketch of Theorem 4 Using developments analogous to those of [NN09c], 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 [SS99] brings in our case:

$$\varepsilon^{0/1}(\boldsymbol{h}^\ell, \mathcal{S}) \leq \frac{1}{C}\sum_{c=1}^{C}\prod_{t=1}^{T} Z_t^{(c)} , \tag{A.9}$$

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:

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

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 (A.9) yields the statement of the Theorem.

- APPENDIX B -

MLNN ALGORITHM

§ B.1 GENERIC MLNN ALGORITHM

Pseudocode of the general version of MLNN is shown in Alg. 6. This is the multi-class version of the same algorithm presented in Appendix A. Indeed, the edge definition is inherently multi-class, and no loop over classes is required while learning the multi-class weak classifiers. As a result, a single leveraging coefficient α_j is learned for each selected prototype. Furthermore, defining the multi-class edge for the use of generic kernels allows us to learn a Multi-class Leveraged Kernel Density classifier, as explained in Chap. 5. Alg. 6 generalizes Alg. 4 of Chap. 5 to the same class of (multi-class) surrogate risks as defined in Appendix A. (See Table A.1.)

§ B.2 MLNN USING EXPONENTIAL LOSS

As pointed out in Sec. 5.2.3, minimizing the multi-class risk function (5.13) amounts to finding δ_j that solves the following equation:

$$-\sum_{i=1}^{m} w_i \mathrm{r}_{ij} \exp\left\{-\delta_j \mathrm{r}_{ij}\right\} = 0\,, \tag{B.5}$$

with: $w_i > 0, \quad \mathrm{r}_{ij} \in \left\{-\frac{1}{(C-1)^2}, \frac{1}{C-1}\right\}$. The following theorem states the existence and uniqueness of the solution for Eq. B.5.

Theorem 5. *For any fixed j, Eq. (B.5) has a finite solution $\delta_j \in \mathbb{R}$ iff there exists a pair of distinct indices $h, l \in \{1, 2, \ldots, m\}$, such that: $\mathrm{r}_{hj} \cdot \mathrm{r}_{lj} < 0$. Moreover, if a solution exists, then it is unique.*

Algorithm 6 MULTICLASS LEVERAGED KERNEL DENSITY CLASSIFICATION $(\mathcal{S}, \psi)$

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

Let $\mathrm{r}_{ij} \doteq \frac{1}{C}\sum_{c=1}^{C} K(\boldsymbol{x_i}, \boldsymbol{x_j}) y_{ic} y_{jc}$ (1)

Let $\alpha_j \leftarrow 0, \quad \forall j = 1, 2, ..., m$

Let $w_i \leftarrow 1/m, \quad \forall i = 1, 2, ..., m$

for $t = 1, 2, ..., T$ **do**

[I.0] Weak index chooser oracle:

Let $j \leftarrow \textsc{Wic}(\{1, 2, ..., m\}, t)$

[I.1] Compute δ_j solution of:

$$\sum_{i=1}^{m} \mathrm{r}_{ij} \nabla_\psi \left(\delta_j \mathrm{r}_{ij} + \nabla_\psi^{-1}(-w_i)\right) = 0\ ; \tag{B.2}$$

$$\sum_{i=1}^{m} w_i \mathrm{r}_{ij} \exp\left\{-\delta_j \mathrm{r}_{ij}\right\} = 0\ ; \tag{B.3}$$

[I.2] $\forall i = 1, 2, \ldots, m$, let

$$w_i \leftarrow -\nabla_\psi \left(\delta_j \mathrm{r}_{ij} + \nabla_\psi^{-1}(-w_i)\right)\ . \tag{B.4}$$

[I.3] Let $\alpha_j \leftarrow \alpha_j + \delta_j$

Output: $h_c^\ell(\boldsymbol{x}_q) = \sum_{t=1}^{T} \alpha_j K(\boldsymbol{x}_q, \boldsymbol{x}_j) y_{jc}\ , \quad \forall c = 1, 2, \ldots, C$

(Proof) Consider the following function $\phi(x)$

$$\phi(x) = -\sum_{i=1}^{m} w_i\, a_i \exp\left\{-a_i x\right\}\ , \tag{B.6}$$

with $w_i > 0$, $a_i \in \mathbb{R}$. Notice that (B.6) is monotonically increasing. Then, for any fixed j and $a_i = \mathrm{r}_{ij}$, $x = \delta_j$, (B.5) is equivalent to:

$$\phi(x) = 0\ . \tag{B.7}$$

Unless to rearrange terms in the summation, we assume without loss of generality that:

$$a_1 \le a_2 \le \cdots \le a_k < 0 < a_{k+1} \le a_{k+2} \le \cdots \le a_m\ , \tag{B.8}$$

and split (B.6) into the sum of negative and positive terms:

$$\phi(x) = \phi^+(x) + \phi^-(x) = -\sum_{i=1}^{k} w_i\, a_i \exp\left\{-a_i x\right\} - \sum_{i=k+1}^{m} w_i\, a_i \exp\left\{-a_i x\right\}\ . \tag{B.9}$$

If $1 \leq k < m$, we have:

- for $x \to -\infty$, $\phi^+(x) \to 0$, $\phi^-(x) \to -\infty$, hence $\phi(x) \to -\infty$
- for $x \to +\infty$, $\phi^+(x) \to +\infty$, $\phi^-(x) \to 0$, hence $\phi(x) \to +\infty$

Therefore $\phi(x)$, which is monotonic, has a zero for some $x \in \mathbb{R}$, thus proving that the condition in theorem's statement is sufficient. Moreover, it is also necessary because, when all a_i's in (B.6) have the same sign, $\phi(x)$ does not change its sign, thus not admitting any zero.

In practical cases, a finite solution is always guaranteed to exist when using a "smooth" kernel that gives rise to a non-sparse edge matrix r_{ij}. For instance, when using the Gaussian kernel (5.31), each column of the edge matrix has both positive and negative entries, thus guaranteeing the finiteness of the solution. However, when the k-NN or truncated kernels are used, column j of the edge matrix may contain only positive (negative) entries, due to its sparsity. In this case, in order to still guarantee to have a finite solution, we propose to modify Eq. B.5 by adding a regularization term, as follows:

$$\sum_{i=1}^{m} \mathrm{r}_{ij}\, w_i \exp\left(-\mathrm{r}_{ij}\delta_j\right) + \frac{\varepsilon}{(C-1)^2}\left[\exp\left(-\frac{\delta_j}{C-1}\right) - \exp\left(\frac{\delta_j}{(C-1)^2}\right)\right] , \tag{B.10}$$

where ε is a small constant, *e.g.*: $\varepsilon = \frac{1}{m}$. This generalizes to generic truncated kernels the setting proposed in Sec. 5.2.4 for the k-NN kernel, which reads as follows:

$$\delta_j = \frac{(C-1)^2}{C} \log\left[\frac{(C-1)w^+ + \varepsilon}{w^- + \varepsilon}\right] . \tag{B.11}$$

When no closed-form solution exists for (B.10), we adapt our Newton iterative scheme for computing the solution numerically.

§ B.3 MLNN USING LOGISTIC LOSS

In this section, we give the specialization of Alg. 6 for the minimization of another widely used loss function, *i.e.*, the logistic loss $\psi^{\log}$. Let us consider the following definition of $\psi^{\exp}$, as provided in Appendix A (Table A.1):

$$\psi^{\log}(x) = \log\left(1 + \exp\left(-x\right)\right) \tag{B.12}$$

Plugging (B.12) into the general MLNN learning equation (B.3), we obtain:

$$\sum_{i=1}^{m} \mathrm{r}_{ij}\, \frac{\tilde{w}_i \exp\left(-\delta_j \mathrm{r}_{ij}\right)}{1 + \tilde{w}_i \exp\left(-\delta_j \mathrm{r}_{ij}\right)} = 0 , \tag{B.13}$$

where we have defined:

$$\tilde{w}_i = \frac{w_i}{1 - w_i} \ . \tag{B.14}$$

In general, (B.13) does not admit a closed-form solution (unless we use the k-NN kernel and all the reciprocal k-nearest neighbors of $\boldsymbol{x}_j$ have equal weights w_i. This latter case corresponds to the particular solution provided in Tab. A.1).

Similarly, by using (B.12) we can write the weight update rule of MLNN in the case of logistic risk minimization:

$$w_i \leftarrow \frac{\tilde{w}_i \exp\left(-\delta_j \mathrm{r}_{ij}\right)}{1 + \tilde{w}_i \exp\left(-\delta_j \mathrm{r}_{ij}\right)} \ . \tag{B.15}$$

In order to solve Eq. (B.13), we implemented a Newton's iterative scheme, similarly as for MLNN with exponential loss (Eq. 5.28).

BIBLIOGRAPHY

[ADB08] Cesario Vincenzo Angelino, Eric Debreuve, and Michel Barlaud. Image restoration using a *k*-NN-variant of the mean-shift. In *IEEE International Conference on Image Processing*, pages 573–576, 2008.

[AGS+10] Uri Avni, Jacob Goldberger, Michal Sharon, Eli Konen, and Hayit Greenspan. Chest x-ray characterization: from organ identification to pathology categorization. In *Multimedia Information Retrieval*, pages 155–164, 2010.

[AL76] Ibrahim A. Ahmad and Pi-Erh Lin. A nonparametric estimation of the entropy for absolutely continuous distributions. *IEEE Transactions on Information Theory*, 22(3):372–375, 1976.

[AMN+98] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of ACM*, 45(6):891–923, 1998.

[AN] A Asuncion and D.J. Newman. UCI machine learning repository. http://archive.ics.uci.edu/ml/.

[AN00] Shun-Ichi Amari and Hiroshi Nagaoka. *Methods of Information Geometry*. Oxford University Press, 2000.

[APPK08] Vassilis Athitsos, Michalis Potamias, Panagiotis Papapetrou, and George Kollios. Nearest neighbor retrieval using distance-based hashing. In *International Conference on Data Engineering*, pages 327–336, 2008.

[ASR06] Jaume Amores, Nicu Sebe, and Petia Radeva. Boosting the distance estimation: Application to the *k*-nearest neighbor classifier. *Pattern Recognition Letters*, 27(3):201–209, February 2006.

[ASS00] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *International Conference on Machine Learning*, pages 9–16, 2000.

[AV07] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Symposium on Discrete algorithms*, pages 1027–1035, 2007.

[AVP+09] Barbara André, Tom Vercauteren, Aymeric Perchant, Michael B. Wallace, Anna M. Buchner, and Nicholas Ayache. Endomicroscopic image retrieval and classification using invariant visual features. In *International Symposium on Biomedical Imaging*, pages 346–349, August 2009.

[AW06] Suyash P. Awate and Ross T. Whitaker. Unsupervised, information-theoretic, adaptive image filtering for image restoration. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 28(3):364–376, 2006.

[BA87] Peter J. Burt and Edward H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, pages 671–679, 1987.

[BA96] Michael J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.

[BCM05] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling and Simulation*, 4:490–530, 2005.

[BDB07] Sylvain Boltz, Eric Debreuve, and Michel Barlaud. High-dimensional statistical distance for region-of-interest tracking: Application to combining a soft geometric constraint with radiometry. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2007.

[Bis07] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[BJM06] Peter L. Bartlett, Micheal I. Jordan, and Jon D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101:138–156, 2006.

[BLSB04] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

[BM02] Henry Brighton and Chris Mellish. Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery*, 6:153–172, 2002.

[BMDG05] Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. Clustering with bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, December 2005.

[BMM07] Anna Bosch, Xavier Muñoz, and Robert Marti. Which is the best way to organize/classify images by content? *Image and Vision Computing*, 25(6):778–791, June 2007.

[BMP02] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 24(4):509–522, April 2002.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networking and ISDN Systems*, 30:107–117, April 1998.

[BSC+99] Ravi Bansal, Lawrence H. Staib, Zhe Chen, Anand Rangarajan, Jonathan Knisely, Ravinder Nath, and James S. Duncan. Entropy-based, Multiple-Portal-to-3DCT registration for prostate radiotherapy using iteratively estimated segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, pages 567–578, 1999.

[BT07] Peter L. Bartlett and Mikhail Traskin. Adaboost is consistent. *Journal of Machine Learning Research*, 8:2347–2368, 2007.

[BZM08] Anna Bosch, Andrew Zisserman, and Xavier Muñoz. Scene classification using a hybrid generative/discriminative approach. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 30(4):712–727, 2008.

[Cay08] Lawrence Cayton. Fast nearest neighbor retrieval for bregman divergences. In *International Conference on Machine Learning*, pages 112–119, 2008.

[Cay09] Lawrence Cayton. Efficient Bregman range search. In *Advances in Neural Information Processing Systems*, pages 243–251, 2009.

[CDF+04] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[CGGR] Stéphane Canu, Yves Grandvalet, Vincent Guigue, and Alain Rakotomamonjy. SVM and Kernel Methods matlab toolbox. http://asi.insa-rouen.fr/enseignants/~arakotom/toolbox/index.html.

[CH67] Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[Cha08a] Sung-Hyuk Cha. Taxonomy of nominal type histogram distance measures. In *American Conference on Applied Mathematics*, pages 325–330, 2008.

[Cha08b] Bernard Chazelle. Technical perspective: finding a good neighbor, near and fast. *Communications of the ACM archive*, 51:115–115, January 2008.

[CRM00] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2000.

[CST00] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[CW02] Yixin Chen and James Z. Wang. A region-based fuzzy feature matching approach to content-based image retrieval. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 24(9):1252–1267, 2002.

[CWK05] Yixin Chen, James Ze Wang, and Robert Krovetz. CLUE: cluster-based retrieval of images by unsupervised learning. *IEEE Transactions on Image Processing*, 14(8):1187–1201, 2005.

[DD09] Thomas Deselaers and Thomas M. Deserno. Medical image annotation in imageclef 2008. In *CLEF Workshop 2008 / Evaluating Systems for Multilingual and Multimodal Information Access*, September 2009.

[DHN10] Thomas Deselaers, Georg Heigold, and Hermann Ney. Object classification by fusing svms and gaussian mixtures. *Pattern Recognition*, 43(7):2476–2484, July 2010.

[DHS01] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, 2001.

[DJ94] David L. Donoho and Ian M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994.

[DKJ+07] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *International Conference on Machine Learning*, pages 209–216, 2007.

[DKN08] Thomas Deselaers, Daniel Keysers, and Hermann Ney. Features for image retrieval: an experimental comparison. *Information Retrieval*, 11(2):77–107, 2008.

[Dud76] Sahibsingh A. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, Cybernetics*, 6(4):325–327, April 1976.

[DV02] Minh N. Do and Martin Vetterli. Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance. *IEEE Transactions on Image Processing*, 11(2):146–158, 2002.

[EGW+] Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[FF84] Keinosuke Fukunaga and Thomas E. Flick. An optimal global nearest neighbor metric. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 6(3):314–318, May 1984.

[FFP05] Li Fei-Fei and Pietro Perona. A Bayesian hierarchical model for learning natural scene categories. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 524–531, 2005.

[FS95] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[FS96] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *ICML*, pages 148–156, 1996.

[GD05] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *IEEE International Conference on Computer Vision*, October 2005.

[GDB08] Vincent Garcia, Eric Debreuve, and Michel Barlaud. Fast k nearest neighbor search using GPU. In *CVPR Workshop on Computer Vision on GPU (CVGPU)*, Anchorage, Alaska, USA, June 2008.

[GLMI05] M. Goria, N. Leonenko, V. Mergel, and P.N. Inverardi. A new class of random vector entropy estimators and its applications in testing statistical hypotheses. *Journal of Nonparametric Statistics*, 17(3):277–297, 2005.

[GPOB09] Nicolás García-Pedrajas and Domingo Ortiz-Boyer. Boosting *k*-nearest neighbor classifier by means of input space projection. *Expert Systems Applications*, 36(7):10570–10582, 2009.

[GPP+07] Lalit Gupta, Vinod Pathangay, Arpita Patra, A. Dyana, and Sukhendu Das. Indoor versus outdoor scene classification using probabilistic neural network. *EURASIP Journal on Applied Signal Processing*, 2007(1):123–123, 2007.

[GRHS04] Jacob Goldberger, Sam T. Roweis, Geoffrey E. Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, 2004.

[GVS09] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Is that you? metric learning approaches for face identification. In *IEEE International Conference on Computer Vision*, pages 498–505, sep 2009.

[HA03] Christopher C. Holmes and Niall M. Adams. Likelihood inference in nearest-neighbour classification models. *Biometrika*, 90:99–112, 2003.

[Har68] P. E. Hart. The Condensed Nearest Neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516, 1968.

[HE03] Greg Hamerly and Charles Elkan. Learning the *k* in *k*-means. In *Advances in Neural Information Processing Systems*, volume 17, 2003.

[HM99] Jinggang Huang and David Mumford. Statistics of natural images and models. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 1541–1547, 1999.

[HT96] Trevor Hastie and Robert Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 18(6):607–616, 1996.

[ima] ImageCLEF (2009). http://ir.shef.ac.uk/imageclef/.

[Jai10] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31:651–666, June 2010.

[JDS10] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2010. to appear.

[JDSP10] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2010.

[JUJ09] V. Jeanne, D. Unay, and V. Jacquet. Automatic detection of body parts in x-ray images. *IEEE Conference on Computer Vision & Pattern Recognition*, 0:25–30, 2009.

[KTZ07] M. P. Kumar, P. H. S. Torr, and A. Zisserman. An invariant large margin nearest neighbour classifier. In *IEEE International Conference on Computer Vision*, 2007.

[KZN08] Neeraj Kumar, Li Zhang, and Shree Nayar. What is a good nearest neighbors algorithm for finding similar patches in images? In *European Conference on Computer Vision*, pages 364–378, 2008.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[LQ65] D. Loftsgaarden and C. Quesenberry. A nonparametric estimate of a multivariate density function. *Annals of Mathematical Statistics*, 36:1049–1051, 1965.

[LSBJ00] Etienne Loupias, Nicu Sebe, Stéphane Bres, and Jean-Michel Jolion. Wavelet-based salient points for image retrieval. In *IEEE International Conference on Image Processing*, 2000.

[LSK+03] Thomas M. Lehmann, Henning Schubert, Daniel Keysers, Michael Kohnen, and Berthold Wein. The irma code for unique classification of medical images. In *International Society for Optical Engineering*, pages 109–117, San Diego, CA, May 2003.

[LSP06] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 2169–2178, 2006.

[LWZZ06] Bin Lin, Xian-Ji Wang, Run-Tian Zhong, and Zhen-Quan Zhuang. Continuous optimization based-on boosting Gaussian mixture model. *International Conference on Pattern Recognition*, 1:1192–1195, 2006.

[MMBG04] Henning Müller, Nicolas Michoux, David Bandon, and Antoine Geissbühler. A review of content-based image retrieval systems in medical applications - clinical benefits and future directions. *International Journal of Medical Informatics*, 73(1):1–23, 2004.

[MMBG06] Oge Marques, Liam M. Mayron, Gustavo B. Borba, and Humberto R. Gamba. On the potential of incorporating knowledge of human visual attention into cbir systems. In *ICME*, pages 773–776, 2006.

[MR09] Bodo Manthey and Heiko Röglin. Improved smoothed analysis of the k-means method. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 461–470, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[MRT09] J.-M. Marin, C.-P. Robert, and D.-M. Titterington. A Bayesian reassessment of nearest-neighbor classification. *Journal of the American Statistical Association*, 2009.

[MS05] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.

[NBN07] Frank Nielsen, Jean-Daniel Boissonnat, and Richard Nock. On Bregman Voronoi diagrams. In *Symposium on Discrete algorithms*, pages 746–755, 2007.

[NLK08] Richard Nock, Panu Luosto, and Jyrki Kivinen. Mixed Bregman clustering with approximation guarantees. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 154–169, 2008.

[NN09a] Frank Nielsen and Richard Nock. Sided and symmetrized bregman centroids. *IEEE Transactions on Information Theory*, 55:2882–2904, June 2009.

[NN09b] Richard Nock and Frank Nielsen. Bregman divergences and surrogates for learning. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 31(11):2048–2059, 2009.

[NN09c] Richard Nock and Frank Nielsen. On the efficient minimization of classification calibrated surrogates. In *Advances in Neural Information Processing Systems*, pages 1201–1208. 2009.

[NS06] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision & Pattern Recognition*, volume 2, pages 2161–2168, June 2006.

[NSJ00] Richard Nock, Marc Sebban, and Pascal Jabby. A symmetric nearest neighbor learning rule. In *European Workshop on Advances in Case-Based Reasoning*, pages 222–233, 2000.

[NWJ09] X. Nguyen, M.-J. Wainwright, and M.-I. Jordan. On surrogate loss functions and f-divergences. *The Annals of Statistics*, 37:876–904, 2009.

[OPM02] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 24(7):971–987, 2002.

[OT01] Aude Oliva and Antonio B. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[PAHD05] E. Pierpaoli, S. Anthoine, K. Huffenberger, and I. Daubechies. Reconstructing sunyaev-zeldovich clusters in future cmb experiments. *Monthly Notices of the Royal Astronomical Society*, 359:261–271, 2005.

[Par06] Roberto Paredes. Learning weighted metrics to minimize nearest-neighbor classification error. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 28(7):1100–1110, 2006.

[PCI+08] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2008.

[PD07] Florent Perronnin and Christopher R. Dance. Fisher kernels on visual vocabularies for image categorization. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2007.

[PRTB99] Jan Puzicha, Yossi Rubner, Carlo Tomasi, and Joachim M. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. In *IEEE International Conference on Computer Vision*, pages 1165–1172, 1999.

[PS05] Andrew Payne and Sameer Singh. Indoor vs. outdoor scene classification in digital photographs. *Pattern Recognition*, 38(10):1533–1545, 2005.

[PSL10] Florent Perronnin, Jorge Sánchez, and Yan Liu. Large-scale image categorization with explicit data embedding. In *CVPR*, pages 2297–2304, 2010.

[PSM10] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the Fisher kernel for large-scale image classification. In *European Conference on Computer Vision*, pages 143–156, 2010.

[PSWS03] Javier Portilla, Vasily Strela, Martin J. Wainwright, and Eero P. Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Transactions on Image Processing*, 12(11):1338–1351, 2003.

[RCB01] J.K. Romberg, H.H. Choi, and R.G. Baraniuk. Bayesian tree-structured image modeling using wavelet-domain hidden markov models. *IEEE Transactions on Image Processing*, 10(7):1056–1068, July 2001.

[Rip94] Brian Ripley. Neural networks and related methods for classification. *Journal of the Royal Statistical Society Series B*, 56:409–456, 1994.

[RMG+76] Eleanor Rosch, Carolyn B. Mervis, Wayne D. Gray, David M. Johnson, and Penny Boyes-Braem. Basic objects in natural categories. *Cognitive Psychology*, 8(3):382–439, 1976.

[RNI09] Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Nearest neighbors in high-dimensional data: the emergence and influence of hubs. In *International Conference on Machine Learning*, pages 865–872, 2009.

[Ros73] Eleanor Rosch. Natural categories. *Cognitive Psychology*, 4(3):328–350, 1973.

[RS02] Saharon Rosset and Eran Segal. Boosting density estimation. In *Advances in Neural Information Processing Systems*, pages 641–648, 2002.

[RSMM00] G. Rätsch, B. Schölkopf, S. Mika, and K.-R. Müller. SVM and Boosting: One class. Technical Report 119, GMD FIRST, Berlin, November 2000.

[SB91] Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[SDI06] G. Shakhnarovich, T. Darell, and P. Indyk. *Nearest-Neighbors Methods in Learning and Vision*. MIT Press, 2006.

[SFBL98] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin : a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26:1651–1686, 1998.

[SHP08] Michael Skalak, Jinyu Han, and Bryan Pardo. Speeding melody search with vantage point trees. In *ISMIR*, pages 95–100, 2008.

[SM83] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[SRE+05] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering object categories in image collections. In *Proceedings of the International Conference on Computer Vision*, 2005.

[SS99] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning Journal*, 37:297–336, 1999.

[SSF+03] Hao Shao, Tomás Svoboda, Vittorio Ferrari, Tinne Tuytelaars, and Luc J. Van Gool. Fast indexing for image retrieval based on local appearance with re-ranking. In *ICIP (3)*, pages 737–740, 2003.

[SSL04] Navid Serrano, Andreas E. Savakis, and Jie-Bo Luo. Improved scene classification using efficient low-level features and semantic cues. *Pattern Recognition*, 37:1773–1784, September 2004.

[SVR05] Eric Spellman, Baba C. Vemuri, and Murali Rao. Using the kl-center for efficient and accurate retrieval of distributions arising from texture images. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 111–116, Washington, DC, USA, 2005. IEEE Computer Society.

[SWS+00] Arnold W. M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, 2000.

[SZ03] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *IEEE International Conference on Computer Vision*, pages 1470–1477, 2003.

[TD10] Tatiana Tommasi and Thomas Deselaers. The medical image classification task. In *Experimental Evaluation in Visual Information Retrieval Series.*, pages 221–238. Springer, 2010.

[TMFR03] Antonio Torralba, Kevin P. Murphy, William T. Freeman, and Mark A. Rubin. Context-based vision system for place and object recognition. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, 2003.

[TOC08a] T. Tommasi, F. Orabona, and B. Caputo. Discriminative cue integration for medical image annotation. *Pattern Recognition Letters*, 29(15):1996–2002, November 2008.

[TOC08b] Tatiana Tommasi, Francesco Orabona, and Barbara Caputo. CLEF2008 Image Annotation Task: an SVM Confidence-Based Approach. Technical report, 2008. CLEF 2008 Working Notes.

[TPJ+09] Yimo Tao, Zhigang Peng, Bing Jian, Jianhua Xuan, Arun Krishnan, and Xiang Sean Zhou. Robust learning-based annotation of medical radiographs. In *Proceedings of the MICCAI Workshop - Medical Content-based Retrieval for Clinical Decision (MCBR-CDS'09)*, volume 5853 of *Lecture Notes in Computer Science*, pages 77–88. Springer, 2009.

[TS92] George R. Terrell and David W. Scott. Variable kernel density estimation. *Annals of Statistics*, 20(3):1236–1265, 1992.

[USE+09] Devrim Unay, Octavian Soldea, Ahmet Ekin, Müjdat Çetin, and Aytül Erçil. Automatic annotation of x-ray images: A study on attribute selection. In *MCBR-CDS*, pages 97–109, 2009.

[VF08] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms, 2008.

[VI97] Paul A. Viola and William M. Wells III. Alignment by maximization of mutual information. *International Journal of Computer Vision*, 24(2):137–154, 1997.

[VS07] Julia Vogel and Bernt Schiele. Semantic modeling of natural scenes for content-based image retrieval. *International Journal of Computer Vision*, 72(2):133–157, 2007.

[WS09] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.

[WWS+06] Zhou Wang, Guixing Wu, Hamid R. Sheikh, Eero P. Simoncelli, En-Hui Yang, and Alan C. Bovik. Quality-aware images. *IEEE Transactions on Image Processing*, 15(6):1680–1689, 2006.

[XHE+10] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 3485–3492, 2010.

[Yia93] Peter N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, SODA '93, pages 311–321. Society for Industrial and Applied Mathematics, 1993.

[YJ06] Liu Yang and Rong Jin. Distance metric learning: A comprehensive survey. Technical report, Department of Computer Science and Engineering, Michigan State University, 2006.

[YJM+10] Liu Yang, Rong Jin, Lily Mummert, Rahul Sukthankar, Adam Goode, Bin Zheng, Steven C. H. Hoi, and Mahadev Satyanarayanan. A boosting framework for visuality-preserving distance metric learning and its application to medical image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, 32(1):30–44, 2010.

[YJZ02] Kai Yu, Liang Ji, and Xuegong Zhang. Kernel nearest-neighbor algorithm. *Neural Processing Letters*, 15(2):147–156, 2002.

[YW10] M. Yuan and M. Wegkamp. Classification methods with reject option based on convex risk minimization. *Journal of Machine Learning Research*, 11:111–130, 2010.

[ZBMM06] Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 2126–2136, 2006.

[ZRZH09] Ji Zhu, Saharon Rosset, Hui Zou, and Trevor Hastie. Multi-class adaboost. *Statistics and Its Interface*, 2:349–360, 2009.

[ZZ07] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

[ZZW08] Wangmeng Zuo, David Zhang, and Kuanquan Wang. On kernel difference-weighted k-nearest neighbor classification. *Pattern Analysis Applications*, 11(3-4):247–257, 2008.