Naive Bayes Image Classification: beyond Nearest Neighbors

Radu Timofte[1], Tinne Tuytelaars[1], and Luc Van Gool[1,2]

[1]ESAT-VISICS /IBBT, Catholic University of Leuven, Belgium
[2]D-ITET, ETH Zurich, Switzerland

Abstract. Naive Bayes Nearest Neighbor (NBNN) has been proposed as a powerful, learning-free, non-parametric approach for object classification. Its good performance is mainly due to the avoidance of a vector quantization step, and the use of image-to-class comparisons, yielding good generalization. In this paper we study the replacement of the nearest neighbor part with more elaborate and robust (sparse) representations, as well as trading performance for speed for practical purposes. The representations investigated are k-Nearest Neighbors (kNN), Iterative Nearest Neighbors (INN) solving a constrained least squares (LS) problem, Local Linear Embedding (LLE), a Sparse Representation obtained by l_1-regularized LS (SR_{l_1}), and a Collaborative Representation obtained as the solution of a l_2-regularized LS problem (CR_{l_2}). In particular, NIMBLE and K-DES descriptors proved viable alternatives to SIFT and, the NBSR_{l_1} and NB$INN$ classifiers provide significant improvements over NBNN, obtaining competitive results on Scene-15, Caltech-101, and PASCAL VOC 2007 datasets, while remaining learning-free approaches (i.e., no parameters need to be learned).

1 Introduction

In [1], Boiman *et al.* introduced a novel, non-parametric approach for image classification, the Naive Bayes Nearest Neighbor classifier (NBNN). Given an image represented by a set of extracted features, and a priori sets/classes of such features with known label, NBNN is searching for the class to which the query features have the minimum cumulated distance. In spite of being a learning-free method and the lack of tuned parameters, this scheme achieves close to state-of-the-art results for image classification. This is due to: i) the lack of a vector quantization step, avoiding thus to introduce more discretization errors, and ii) the use of image-to-class comparisons instead of image-to-image, which allows to combine bits and pieces of information from multiple training images.

At the same time, NBNN also has its limitations. The most important one is the high computational cost during testing: computation time depends linearly on the number of extracted features from the query image and linearly to logarithmicly on the number of stored labeled features. Unfortunately, the performance heavily depends on the density of features: the more information, the better the results but the slower the evaluation. Some have also criticized the Naive Bayes learning free approach, as it needs balanced data with similar density in feature space for all classes, which often is not the case, and also the feature independence assumption is questionable. Learning is the solution to these problems, as addressed by [2–4]. Here we do not follow this strand of work, but rather stick to the simplicity of the original, parameter-free NBNN based on Naive Bayes. Instead, we explore another, complementary direction for improving the results, by investigating alternative representations that can replace the Nearest Neighbor scheme. We believe it is easier to spot the relative power of different representations in a learning-free formulation. Adapting the learning-based extensions of NBNN of [2–4] to the Naive Bayes variants investigated here seems relatively straightforward.

NBNN starts from the basic Naive Bayes (NB) Classifier, and uses the distance to the Nearest Neighbor (NN) as an approximation of the log-probability for a single feature to belong to a certain class of features. This is a somewhat arbitrary choice, yet critical for the overall performance. Arguably, NN is not the best option. In sparse coding approaches for object classification [5], NN in the form of so-called 'hard assignment' was proven inferior to the 'soft-assignment' approaches such as l_1-regularized least squares. Moreover, the NN classifier has considerably lower performance than state-of-the-art classifiers such as Support Vector Machines or Sparse Representation-based Classifiers (SRC) for the recognition of faces [6], handwritten digits or traffic signs [7].

Replacing the NN with a different representation in the NB classifier formulation is not affecting the desirable properties of NBNN: i) discretization is still avoided, ii) the good generalization of 'image-to-class' is maintained, and iii) it is still a learning free method. The parameters of the representations are adjustable, but usually have known general values for which the performance is reasonably high on (largely) different features and settings.

The main contributions of this paper are as follows. We investigate more general variations of the Naive Bayes Classifier starting from sparse representations. We empirically evaluate k-Nearest Neighbors (kNN), Local Linear Embedding (LLE) [8], Iterative Nearest Neighbors (INN) [9], Sparse Representation with l_1-regularization (SR_{l_1}) [6], and the Collaborative Representation with l_2-regularization (CR_{l_2}) [10]. Except for the last one, all of these are sparse. Moreover, we use in the NB framework for the first time the recently proposed Kernel Descriptors [11] and compare them to the results obtained with NIMBLE features [12].

Another side contribution is the evaluation of speeding up strategies for Naive Bayes Classification. We show that with only a small overhead over the original NBNN introduced by computing the rich representations, we are able to improve the performance. Moreover, we evaluate schemes with asymmetric sampling density as in [4] and approximated nearest neighbors (ANN) [13], with respect to the impact on performance and the speedup achieved.

Section 2 of this paper reviews the NB classifier and the sparse representations based variants. Section 3 gives a time complexity analysis. Section 4 describes our experimental results. Section 5 discusses trade-offs between performance and speed for practical applications, and Section 6 concludes the paper.

2 Naive Bayes classification

The NB classifier [1] is a probabilistic classifier working under strong independence assumptions. A query image is represented by a set $Q = \{q\}$ of features which are assumed to be independently sampled from a class-specific feature distribution $p(q|c)$. Assuming uniform priors $p(c)$, the classification then becomes

$$\begin{aligned} \hat{c} &= \arg\max_{c\in C} p(c|Q) &&= \arg\max_{c\in C}\{\textstyle\prod_{q\in Q} p(q|c)\} \\ &= \arg\min_{c\in C} -\log\{\textstyle\prod_{q\in Q} p(q|c)\} &&= \arg\max_{c\in C}\{\textstyle\sum_{q\in Q}\log p(q|c)\} \end{aligned} \tag{1}$$

In the following we review some of the most popular techniques for defining a (sparse) representation for a query from a given set of samples and we show how the Naive Bayes Classifier can be adapted to use these representations.

2.1 k Nearest Neighbors (kNN)

kNN is a standard method for defining the locality of a sample. The parameters required are the similarity or distance measure necessary for ordering the neighbors and the number of nearest neighbors to be kept. Boiman *et al.* [1] have shown impressive results using NB and $1NN$ for image classification.

In the case of NB-NN_k, for a query image represented by its set of features $Q = \{q\}$, we compute the k-nearest neighbors from each class $c \in C$, notated $N_k^c(q)$, and the (squared) distance-to-class d_q^c and the classification becomes

$$\hat{c} = \arg\min_{c\in C} \sum_{q\in Q} d_q^c, \qquad d_q^c = \sum_{x\in N_k^c(q)} \|q - x\|^2 \tag{2}$$

which corresponds to the class with the minimum cumulated distances to the query. It is assumed that $p(q|c) \sim \exp(-d_q^c)$. Here, we consider only neighborhood sizes: $k = 1$, *i.e.* NBNN_1, which is the original NBNN [1], and $k = 5$, referred to as NBNN_5.

2.2 Local Linear Embedding (LLE)

LLE [8] was introduced as a non-linear dimensionality reduction method, and recently incorporated in the Locality-constrained Linear Coding (LLC) method [14].

For a given sample q and the set $N_k(q)$ of kNN from *all* the labeled samples X, LLE solves:

$$\min_w \|q - \textstyle\sum_{x\in N_k(q)} w_x x\|^2 \,, \qquad \text{subject to } \|w\|_1 = 1 \tag{3}$$

where w is a k-dimensional weights vector corresponding to the kNN, with its l_1-norm constrained to 1. In our experiments, we consider $5NN$ as in LLC [14].

The importance of each sample $x \in N_k(q)$ in the reconstruction of q is given by the absolute value of the assigned weight, $\tilde{w}_x = |w_x|$, while the non-neighboring samples, $x \notin N_k(q)$, get $\tilde{w}_x = 0$. We further l_1-normalize the importance weights for a sample q to sum to 1, yielding $\hat{w}^q = \tilde{w}/\|\tilde{w}\|_1$. Now the *likeliness* of q to belong to class c is given by:

$$d_q^c = \sum_{x \in X_c} \hat{w}_x^q \tag{4}$$

where X_c is the subset of X corresponding to the class c.

The decision for the NB$LLE$ classifier gives the class with the largest importance/likeliness in locally linear embedding the query image Q:

$$\hat{c} = \arg\max_{c \in C} \sum_{q \in Q} d_q^c \tag{5}$$

2.3 Sparse Representation with l_1-min (SR_{l_1})

The power of l_1-regularized least squares or *lasso* is proven for problems such as face recognition [6] or dimensionality reduction [7]. Given the set of M labeled samples $X = \{x\}$, a matrix is formed $\mathbf{X} = [x_1, x_2, \dots, x_M]$. Then, for a given sample q, we optimize over w (we fix $\lambda = 0.3$):

$$\min_w \|q - \mathbf{X}w\|_2 + \lambda\|w\|_1 \tag{6}$$

For capturing the importance of the weights for the query q, we again take their absolute value $\tilde{w} = |w|$, as in [7], and further l_1 normalize them as in the case of LLE, yielding $\hat{w}^q = \tilde{w}/\|\tilde{w}\|_1$. Now the likeliness d_q^c of q to belong to class c and the decision for the NBSR_{l_1} classifier is taken similarly to NB$LLE$ (eqs. (4)(5)) and selects the class that scores the largest importance/likeliness in the sparse representations of the query image Q. In this form NBSR_{l_1} has been used in [9].

2.4 Collaborative Representation (CR_{l_2})

CR_{l_2} [10] seen as l_2-regularized least squares can be as good as the SR_{l_1} on face recognition under specific conditions, like high dimensional face representations and suitable features such as eigenfaces. A strong point of this approach is the existence of an algebraic solution for

$$\min_w \|q - \mathbf{X}w\|_2^2 + \lambda\|w\|_2^2 \tag{7}$$

(we fix $\lambda = 0.001$) such that the coding of q over X is given by

$$w = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T q \tag{8}$$

For a specific class c the regularized residuals are computed, as in [10]:

$$r_c^q = \|q - \mathbf{X}_c w_c\|_2 / \|w_c\|_2 \tag{9}$$

where w_c is the (sub)vector of coefficients corresponding to the class c samples from the whole representation w. Similarly $\mathbf{X}_c$ is the (sub)matrix of $\mathbf{X}$ containing only the samples corresponding to class c. The decision for NBCR_{l_2} is then

$$\hat{c} = \arg\min_{c \in C} \sum_{q \in Q} r_q^c \tag{10}$$

corresponding to the class with minimum cumulated residuals to the query Q.

2.5 Iterative Nearest Neighbors (INN)

INN [9] representation is an approximate solution to a LS decomposition with imposed coefficients/weights and optimized selected samples. INN tries to fill the gap between fast kNN approaches and powerful but slow sparse or collaborative representations (SR_{l1},CR_{l2}). Given the set of labeled samples $X = \{x\}$, k samples s_i are picked iteratively optimizing

$$\arg\min_{\{s_i\}_{i=1}^K} \|q - \sum_{i=1}^{K} \frac{\lambda}{(1+\lambda)^i} s_i\|_2, \quad K = \lceil -\frac{\log(1-\beta)}{\log(1+\lambda)} \rceil \tag{11}$$

where λ is set to 0.1 and β is 0.95 as in [9], thus $K = 25$. Since the weights are nonnegative and sum up to 1, the likeliness of q to belong to class c is taken as:

$$d_q^c = \sum_{s_i \in X_c} w_{s_i}^q = \sum_{s_i \in X_c} \frac{\lambda}{(1+\lambda)^i} \tag{12}$$

where X_c is the subset of X corresponding to the class c and s_i is the i-th selected sample in the INN representation.

The decision for the NB$INN$ classifier is taken similarly to NB$LLE$ (eq. (5))

3 Time analysis

3.1 Time complexity analysis

NBNN_k depends on the number of features from the query image, $N = |Q|$, the number of labeled features, $M = |X|$, the dimensionality of the features, D, and the number of nearest neighbors, k. The time complexity of a straightforward implementation is $O(NMD+NMk)$. Using the structure of the known data, and organizing it with standard data structures such as kd-trees or hashing tables, allows for a practical sub-linear search in M for k nearest neighbors, without loss of accuracy. However, the higher the dimensionality of the features, the lower the speedup achieved, to the point of no gain.

NB$LLE$ has a time complexity of $O(NMD + NMk + NMk^2D)$, where $O(NMk^2D)$ is the added time complexity introduced by solving the local linear embedding over k-nearest neighbors. In most practical cases $k \ll D$, thus NB$LLE$ is k^2 times slower than NBNN_k.

NB$INN$ is running iteratively NN, thus its time complexity is the number K of iterations, equation (11), times the time complexity of NBNN_1, $O(NMDK)$.

NBSR_{l_1} greatly depends on the l_1-minimization solver employed for solving equation (6). The feature sign algorithm solver used by us [15] has a time complexity, in the optimal case, of $O(MDs)$, where s depends on the non-zero coefficients in the solution. The solutions tend to be very sparse, so s is small. The time complexity of NBSR_{l_1} is $O(NMDs)$ and, still, much slower than NBNN_k.

For NBCR_{l_2}, first computing the projection matrix of equation (8) greatly reduces the further computation time. The time complexity for this is $O(M^2D+M^3+NMD)$, so depending heavily on the number of labeled features, M. If one uses the pseudoinverse [16] then we have $O(D^3+D^2M+NMD)$.

With naive implementations all the NB variants presented are adding time complexity to that of the original NBNN_1 algorithm. Usually, in our settings, the running times are, from the fastest to the slowest, in the following order: NBNN_1 <NBNN_k <NB$LLE$ ≈NB$INN$ ≪NBSR_{l_1} <NBCR_{l_2}, where NBSR_{l_1} and NBCR_{l_2} can be orders of magnitude slower than the NBNN_1 classifier.

3.2 Approximations analysis

At first glance, the presented NB variants are doomed to unfeasible computational burden. Luckily, all methods can be speeded up significantly using a number of approximations, as discussed next. First, as used in [14], LLE can be seen as a local approximation of the richer LLC, which instead of kNN is defined over the whole data. Also for SR_{l_1} the performance is seen to degrade slowly when the labeled features used for the decomposition are reduced to a local neighborhood of sufficient size. In [5] only the top nearest neighbors are kept, reducing the computational burden and causing a drop in time complexity to $O(MD+Mk+kDs)$ for solving a single l_1-minimization. The same trick can be applied for CR_{l_2}, so the critical part remains the retrieval of the nearest neighbors.

ANN. The computational burden is now finding the k-nearest neighbors where k is either 1 (NBNN_1 and NB$INN$), 5 (NBNN_5 and NB$LLE$), or 200 (NBSR_{l_1} and NBCR_{l_2}). A lot of effort has been devoted to sub-linear searching by using the structure of the data, and accurate techniques such as kd-tree or hashing can be used especially when the feature dimensionality is low. In our case, the speedup over linear search is small with such approaches. Therefore, we employ approximated nearest neighbors. Here, we use the FLANN implementation [13]. At the price of lower accuracy (out of k-nearest neighbors, on average only 90% are accurately retrieved), we gain a substantial speedup. ANN was also used in [1, 4].

Asymmetry. A further improvement in running time of the NB variants can be obtained by sampling less densely. We experiment with an asymmetric scheme where we lower the sampling density on the training material (reducing M), or on the testing material (reducing N). Usually, the speedup obtained is equal to the subsampling ratios (except if combined with ANN, when it is significantly lower than the subsampling ratio in training material). [3] suggested reducing the sampling density and learning a metric in feature space to compensate for the loss in accuracy. [4] shows that reducing the sampling density in the query usually brings smaller than expected accuracy loss. We arguably show that it is better to keep as dense as possible the features on the query, exploiting all the information available in the query image. Reducing the sampling density for the known labeled features affects less the performance of the NB variants than sampling less densely on the query image. The reason behind this is: if the set of labeled features is sufficiently large, then it is likely to contain redundant data, and subsampling these features still keeps the overall distribution, while subsampling the query image results in loss of valuable data for classification.

4 Experimental results

4.1 Feature extraction

In our experiments we use both NIMBLE [12] and dense Gradient Kernel Descriptors (G-KDES) [11]. This choice is motivated by the reported performance of both features, consistently outperforming the more conventional SIFT [17] features. Moreover, the NIMBLE features are richer and achieve higher reported performance than SIFT while being an order of magnitude sparser. For NIMBLE we use the code of [12] and follow exactly the settings as in their paper (100 fixations per image and 500-dimensional feature descriptors). Due to its sparse extraction, the running time of the NB methods with NIMBLE is considerably lower than in the case of dense G-KDES or dense SIFT and it is therefore the first choice for most of the experiments we perform. For G-KDES we follow the settings from [11]: extracting dense G-KDES features over a regular grid with 8 pixels step and 16×16 pixels patches, after resizing the image to a maximum of 300×300 pixels. We further lower the dimensionality through PCA to 64.

We include the weighted relative image coordinates in our feature descriptors, as suggested in [1]. Without them there is a significant drop in performance for NBNN [4]. The image coordinates are normalized to $[0, 1]$ by dividing with the width and height of the image and added with an empirically fixed weight of $\sqrt{0.5}$ to the l_2-normalized feature descriptors. This is equivalent to the value 0.5 used in [12] to weight the square l_2 distances.

4.2 Implementation

In the case of LLE, we pick the $5NN$. This is motivated by [14], where the LLC framework performs best when the neighborhood size is 5. For solving (6) for the SR_{l_1} representation, we use the feature sign search algorithm [15] run only on the top NNs to obtain approximated sparse coefficients. However, if the ratio of nearest neighbors *w.r.t.* the total number of samples used for solving an approximated SR_{l_1} representation is too small the loss in performance is significant [9]. Thus, we use 200-NN for the experiments up to 10000 training samples, 500-NN for up to 50000 and 2000-NN for up to 150000. Similarly to [5], for the feature

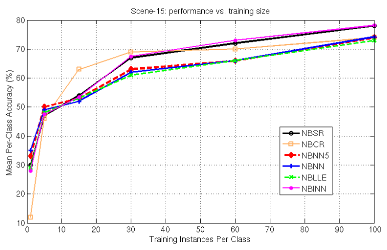

Fig. 1. Performance vs. training size on Scene-15.

sign algorithm, the parameter λ is set to 0.3, which forces a very sparse representation. In case of CR_{l_2}, equation (7), we first compute the projection matrix, thus speeding up the subsequent computations. For the experiments with ANN we use [13] and build the data structure offline with a target of 90% correct neighbors in a k-NN query. Note that we are building one data structure for the whole pool of labeled features and we retrieve only once the 200-NN for each query feature. From these we pick the k-NN, for NBNN_k, for each class and, in absence of representatives, we take the largest distance to the retrieved samples. This approach was taken also in [9, 18]. In [18] the neighborhood size is tuned for best performance with FLANN, calling this method Local NBNN.

4.3 Scene-15

Scene-15 is a popular benchmark from [19]. We follow the common experimental settings, with a training partition of 100 images per class. We generate 20 random train/test splits and for each keep the performance as the average over per class accuracies. Finally we compute mean performance and standard deviation.

Performance vs. training size. First, we evaluate the impact of the number of training images per class on the performance using NIMBLE features (see Fig. 1). From Fig. 1 we see that considering more neighbors in the NB decision does not necessarily improve the performance. NBNN_1, NBNN_5 and NB$LLE$ perform similarly. NBCR_{l_2} is the best for 10 to 40 training images, while NBSR_{l_1} takes the lead after 40 images. NB$INN$ is on par with NBSR_{l_1}.

Performance vs. feature density. In Table 1 we report the accuracies for the NB variants under different uniform sampling densities of NIMBLE features in training and test images. Also the effect of using ANN is evaluated. We report mean classification rates and standard deviation for 20 trials for each setting. The best performance is achieved when sampling with the highest possible density in both training and testing images (78.2%-NBINN,77.8%-NBSR_{l_1},74.3%-NBNN_5,74.2%-NBNN,73.7%-NBCR_{l_2},73.1%-NBLLE). Reducing the density results in some performance loss, yet significantly speeds up all methods. Using only a single NIMBLE feature per image, the accuracy for NBSR_{l_1} and NBCR_{l_2} is still above 30% and 35%, respectively. This is mostly due to the saliency guided NIMBLE feature extraction and the richness of the descriptor itself. The richer representations greatly improve over the standard NN_1 espe-

Table 1. Performance versus asymmetry in train/test and use of ANN on Scene-15.

#tr/#te	With asymmetry							With ANN(target 90%) and asymmetry				
features	speedup	NBNN_1	NBNN_5	NBCR_{l_2}	NBSR_{l_1}	NB$INN$	NB$LLE$	speedup	NB$NN$	NBSR_{l_1}	NB$INN$	NB$LLE$
1/1	10000x	26.0 ± 1	33.8 ± 1	35.7 ± 1	30.7 ± 1	29.8 ± 1	28.5 ± 1	10000x	25.92 ± 1.3	30.63 ± 0.9	29.13 ± 1.7	28.49 ± 0.8
5/5	400x	52.2 ± 1	59.8 ± 1	63.2 ± 1	58.8 ± 1	58.2 ± 1	53.1 ± 1	500x	52.19 ± 1.0	58.83 ± 1.2	56.95 ± 1.9	53.05 ± 0.6
10/10	100x	62.3 ± 1	65.6 ± 1	68.3 ± 1	68.7 ± 1	68.4 ± 1	62.5 ± 1	200x	61.26 ± 1.1	66.29 ± 1.1	65.86 ± 1.0	61.37 ± 1.2
20/20	25x	67.0 ± 1	70.6 ± 1	70.0 ± 1	71.1 ± 1	71.6 ± 1	66.4 ± 1	75x	66.86 ± 1.0	71.44 ± 0.8	70.61 ± 1.5	66.45 ± 0.7
50/50	4x	72.1 ± 1	74.1 ± 1	72.2 ± 1	75.5 ± 1	76.2 ± 1	70.5 ± 1	12x	71.30 ± 0.7	74.33 ± 0.7	74.18 ± 1.4	69.00 ± 1.8
100/100	**(REF)**	**74.2±1**	**74.3±1**	**73.7±1**	**77.8±1**	**78.2±1**	**73.1±1**	5x	73.23 ± 1.0	77.26 ± 1.2	77.65 ± 1.6	68.33 ± 1.2
100/50	2x	71.2 ± 1	70.9 ± 1	71.8 ± 1	77.1 ± 1	77.6 ± 1	71.0 ± 1	10x	70.56 ± 1.1	76.59 ± 1.2	76.72 ± 1.4	67.73 ± 0.7
100/20	5x	67.4 ± 1	68.8 ± 1	70.5 ± 1	75.3 ± 1	76.7 ± 1	67.6 ± 1	25x	66.68 ± 0.9	74.84 ± 1.1	74.93 ± 1.5	63.01 ± 0.9
100/10	10x	65.5 ± 1	66.5 ± 1	69.4 ± 1	72.7 ± 1	73.1 ± 1	63.8 ± 1	50x	65.09 ± 1.2	71.97 ± 1.0	71.45 ± 1.4	58.28 ± 1.2
100/1	100x	39.5 ± 1	40.7 ± 1	58.4 ± 1	54.6 ± 1	55.2 ± 1	42.8 ± 1	500x	39.09 ± 1.3	54.32 ± 1.5	54.13 ± 1.0	41.16 ± 1.4
1/100	100x	65.0 ± 1	65.7 ± 1	65.1 ± 1	66.4 ± 1	66.1 ± 1	64.5 ± 1	100x	64.73 ± 1.3	66.16 ± 1.4	65.81 ± 1.6	64.32 ± 1.6
10/100	10x	71.1 ± 1	72.9 ± 1	72.0 ± 1	73.4 ± 1	72.8 ± 1	71.5 ± 1	20x	70.68 ± 0.7	73.18 ± 1.3	72.50 ± 1.2	70.40 ± 1.2
20/100	5x	72.6 ± 1	74.4 ± 1	70.8 ± 1	75.5 ± 1	75.0 ± 1	72.1 ± 1	15x	72.53 ± 1.1	75.06 ± 1.0	74.64 ± 1.1	71.77 ± 0.9
50/100	2x	72.7 ± 1	72.3 ± 1	73.5 ± 1	76.6 ± 1	76.9 ± 1	72.3 ± 1	6x	72.27 ± 0.7	75.83 ± 0.9	75.17 ± 1.3	71.13 ± 1.2

cially in the lower densities settings. Also note how for lower densities, NBNN_5 also does improve over the NBNN_1.

In an asymmetric setting, the performance decreases much faster when lowering the density in the query as compared to lowering the density for the training images, especially for NB$NN$. With only one feature per query image, NBCR_{l_2} and NBSR_{l_1} still give a reasonable performance (58% and 55%) vs. NBNN_1 (39%). In the reversed case, with one feature per training image, the drop in performance is much lower for all the methods, and NBCR_{l_2} / NBSR_{l_1} still reach 65%/ 66%. NB$INN$ is on par with NBSR_{l_1} for all the settings.

Performance vs. running time. Note that subsampling can result in enormous speedups, with often acceptable drops in performance. For instance, with NBSR_{l_1} a 100× speedup is achievable at the price of a 9.1% drop in performance (68.7% vs. 77.8%). NB$INN$ is 10× faster than and on par with NBSR_{l_1}. The ANN speedup increases with the size of the training, and decreases with feature dimensionality. Relaxing the target, by allowing more incorrect neighbors, lowers the performance but the gain in speed might pay off. One NB$NN$ experiment on Scene-15 using ANN and 100 NIMBLE features per image, takes less than 3 hours on a 2009 Core 2 Quad machine. NBSR_{l_1} is slower, requiring about 4 hours, using a similar 200-NN retrieval per each feature. Nevertheless, the running time per query image per class is way below 1 second, which corresponds to the NBNN time reported in [1] (using dense SIFT).

State-of-the-art. In Table 2 we compare the performance of the NB classifiers proposed here to state-of-the-art results reported in the literature. All these top methods are learning based, i.e. they require prior training and parameter estimation. The Naive Bayes methods, on the other hand, are learning-free and parameter-free to a large extent. One could tune λ for SR_{l_1} in equation (6), hoping in a better fit to the feature space used, but we did not try this. NBSR_{l_1} and NB$INN$ methods improve over the standard NBNN_1 using either NIMBLE or G-KDES features. The gain in performance is more than 4% using INN representations. The recently proposed, Kernelized NBNN with dense SIFT [4], (a learning based method), has similar performance with the NBSR_{l_1} and NB$INN$ methods. Our best performance is achieved by NBSR_{l_1} with G-KDES features and is close in performance to the standard learned method Sparse Coding Spatial Pyramid Matching (78.7% vs. 80.3%). Note that kernelizing our methods and

Table 2. Performance Comparison on Scene-15 and Caltech-101

Scene-15				**Caltech-101**			
	Method	100 images			Method	15 images	30 images
Learned	ScSPM+SIFT [5]	80.28 ± 0.93		Learned	ScSPM+SIFT [5]	67.0 ± 0.5	73.2 ± 0.5
	EMK+KDES [11]	87.5			EMK+KDES [11]	?	77.5
	LScSPM+SIFT [22]	**89.75±0.50**			GLP [21]	**70.34**	**82.6**
	NBNN&BoF+SIFT [4]	85 ± 4			NBNN&phow+SIFT [4]	69.2 ± 0.9	75.2 ± 0.4
	NBNN-f_2+SIFT [4]	79 ± 2			LLC+HOG [14]	65.43	73.4
Learning-free	NB-NN+NIMBLE	74.2 ± 1		Learning-free	NB-NN+NIMBLE	70.1 ± 1	78.1 ± 1
	NB-NN_5+NIMBLE	74.2 ± 1			NB-NN_5+NIMBLE	70.2 ± 1	78.2 ± 1
	NB-SR_{l_1}+NIMBLE	**77.8±1**			NB-SR_{l_1}+NIMBLE	**71.8±0.8**	**79.73±1.1**
	NB-INN+NIMBLE	**78.2±1**			NB-INN+NIMBLE	**72.1±1.2**	**80.29±1.0**
	NB-CR_{l_2}+NIMBLE	73.7 ± 1			NB-CR_{l_2}+NIMBLE		
	NB-LLE+NIMBLE	74.0 ± 1			NB-LLE+NIMBLE	70.4 ± 1	78.2 ± 1
	NB-NN+G-KDES	75.1 ± 1			NBNN+SIFT [1]*	65.0 ± 1.1	70.4
	NB-NN_5+G-KDES	75.1 ± 1			NBNN+NIMBLE [12]*	70.8 ± 0.7	78.5 ± 1.2
	NB-SR_{l_1}+G-KDES	**78.7±1**			LocalNBNN+SIFT [18]*	66.1 ± 1.1	71.9 ± 0.6
	NB-INN+G-KDES	**79.8±1**					
	NB-CR_{l_2}+G-KDES	74.5 ± 1					
	NB-LLE+G-KDES	76.4 ± 1			* indicates results without background class		

combining them with bag-of-features based methods as in [4] is likely to increase our results further (at the cost of switching to a learning-based scheme).

4.4 Caltech-101

On Caltech-101 [20] we report results both with NIMBLE and G-KDES features. In our evaluation (see Table 2), the performance of the NB$NN$ classifier with NIMBLE features compares to the one reported in the original paper [12] (note that their result is without considering the background class). All the NB variants achieve comparable or better results than the state-of-the-art. 70.3% at 15 images per class and 82.6% at 30 images is the best performance out of the state-of-the-art learning methods, achieved by the GLP method of [21]. NBSR_{l_1} with NIMBLE features reaches 71.8%@15 and 79.73%@30, and NB$INN$ with NIMBLE features reaches 72.1%@15 and 80.29%@30. This is the best result using a single descriptor for Caltech-101 with 15 train images per class, to the best of our knowledge.

4.5 PASCAL VOC 2007

The PASCAL VOC 2007 [23] has a much higher variability in shape, pose, and position for the 20 annotated object classes than Caltech-101 or Scene-15. We are using NIMBLE features and report results with different sampling densities in training and test images respectively. We report for NB$INN$ the results from [9] and run only NB$NN$ and NBSR_{l_1} since the other NB variants gave lower, less robust performance on the previous datasets.

For this challenge, we need to provide class confidences. The relative ranking among the classes for a specific image query is not sufficient, as some images contain instances of multiple classes. Using directly the score from equation (2) for NB$NN$ is not beneficial. The scores need to be brought to the same meaning, to be comparable across the image queries and not just for the class decision. To this end, we consider the likelihood ratio between the probability of belonging to the class and the probability of not belonging to the class. This resembles the f_2 kernel from [4], and gives the following confidence function:

$$S = \sum_{q \in Q} (d_q^c - d_q^{\bar{c}}), \qquad d_q^{\bar{c}} = \sum_{x \in N_k^{\bar{c}}(q)} \|q - x\|^2 \tag{13}$$

Table 3. Image classification results using PASCAL VOC 2007 dataset

object class + #tr/#te	aero	bicyc	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	average
Best of VOC'07[23]	77.5	63.6	56.1	71.9	33.1	60.6	78.0	58.8	53.5	42.6	54.9	45.8	77.5	64.0	85.9	36.3	44.7	50.9	79.2	53.2	59.4
NBINN[9]	69.8	63.8	48.5	61.9	26.6	58.9	74.8	52.9	51.6	36.0	42.5	40.8	75.7	62.2	82.8	22.1	28.9	41.3	74.1	46.0	53.1
INNSPM[9]	77.2	64.4	56.2	71.4	32.7	69.1	80.0	59.8	49.5	47.9	55.3	45.8	77.8	67.0	84.6	30.2	44.7	53.4	79.1	53.8	**60.0**
NBNN+1/100	57.1	34.3	17.3	25.0	07.5	31.8	51.3	32.1	27.6	16.3	19.2	19.7	57.6	37.3	50.5	10.5	10.9	24.1	48.4	22.4	30.4
NBNN_S+1/100	62.3	39.0	23.8	38.6	13.1	33.6	64.9	36.8	33.5	21.5	31.0	27.5	61.2	47.4	69.5	18.2	17.1	25.5	53.0	28.2	37.3
NBNN_S+10/10	61.7	43.2	24.9	30.5	11.2	31.5	61.8	32.1	31.5	16.3	18.1	26.3	60.3	39.2	66.4	11.4	17.8	22.2	51.6	26.0	34.2
NBNN_S+100/100	69.4	59.5	39.3	45.7	22.3	54.2	73.9	49.6	42.7	29.2	39.8	34.7	72.7	61.3	76.6	13.8	27.1	33.8	71.5	42.6	48.0
NBNN_S+100/1000	70.7	59.3	40.3	47.4	23.0	57.4	74.3	50.7	42.2	32.9	43.7	35.7	72.9	61.8	78.1	14.5	29.1	34.8	73.1	44.9	**49.3**
NBSR_{l_1}+1/1	36.9	17.1	09.2	13.7	04.8	14.4	32.0	11.2	21.0	04.4	14.2	16.0	18.2	16.5	52.7	06.3	02.9	09.2	17.6	08.4	16.3
NBSR_{l_1}+10/10	62.0	47.2	28.4	41.7	12.4	35.7	66.1	36.7	37.9	18.8	20.3	28.8	62.8	45.0	72.6	09.7	17.7	24.4	57.7	32.8	37.9
NBSR_{l_1}+100/100	67.9	63.2	46.1	50.7	21.6	56.6	75.2	54.5	46.4	34.2	40.9	41.4	73.5	62.1	81.4	21.1	26.9	41.9	72.7	44.9	51.2
NBSR_{l_1}+100/1000	67.9	63.6	47.0	50.5	22.4	57.9	75.3	56.0	47.0	34.6	37.9	41.6	74.8	62.5	81.7	21.4	28.1	43.5	73.1	46.2	**51.7**

for a given query Q and a class c, where $\bar{c}$ is the negative class. The NB$NN$ method with normalized scores is noted NBNN_S, where $k = 1$ as in NB$NN$.

In Table 3 we compare our different settings with the best results from the challenge. When using a strong asymmetry (just 1 feature per training image, 100 features per query image), we already achieve a mean average precision of 37.3% for NBNN_S. Using equal densities (100/100) we obtain a performance of 48.1%, which is 11% behind the best entry in the VOC2007 challenge. The sparse representations pay off and the NBSR_{l_1} improves over NBNN_S resulting in a mean average precision of 52% or only 7% below the top results. This is a good result, taking into account that the best entry in VOC2007 is a learning-based method, more complex than our learning-free approaches. Moreover, we used only the top 200NN to compute the NBSR_{l_1} decision, while NB$INN$ uses the whole training pool.

5 Discussion

Computational costs. The main limitation of NB classifiers is the high computational cost. As shown in Section 3 the time complexity depends (usually) linearly on the number of training images, feature extraction density in training and testing, and feature dimensionality. All these made the deployment of NB$NN$ prohibitive in practice. It remained just a theoretical exercise. Even with exploiting the data structure and using ANN, NB$NN$ is quite slow in traditional settings, *i.e.* for $\sim 5,000$ SIFT features per image and 3,000 training images, it takes $\sim$ 1 second per query image and class [1]. Here, we show how to temper the computational costs for NB schemes so they can be used in practice.
Features. The features are the building blocks in NB image classification. Ideally, features should be statistically significant for classification with a low dimensional representation and a low sampling density. The main stream literature [1–4] heavily relies on the 128 dimensional SIFT features that for high classification performance require uniform sampling densities above 3,000 features per normalized < 0.1 mega pixels images. The recently proposed NIMBLE [12] features (500 dimensional, 100 sampling density) and G-KDES [11] features (200 dim., < 1000 density) proved to improve over SIFT based classification and come with (much) lower sampling densities. Using NIMBLE features gives two orders of magnitude speedup over the NBNN SIFT baseline. G-KDES features are also able to severely reduce (one order of magnitude) the running time of a NBNN

SIFT baseline. The Matlab implementations as provided by [12, 11] are running in the range of seconds and can be optimized further.

Training size. The number of training images per class heavily affects the performance. The larger their number the higher the classification rate but at the price of an increase in run time. From Fig. 1 we observe that for up to 5 training images per class, NBCR_{l_2} is the worst performing method, while NBNN_1 and NBNN_5 are best. Collaborative Representation (CR_{l_2}) shows clearest benefit in performance in the middle range (10 up to 40 training images), while after 40 training images NBCR_{l_2} performance scales less well *w.r.t.* the other NB variants. When more than 40 training images per class are available, NBSR_{l_1} and NB$INN$ are clearly the best choices. To the limit, when infinite number of training images are available, it is expected that all NB variants with the exception of NBCR_{l_2} converge in performance. A topic of further research is: how to filter the training features without loss in performance, *e.g.* by feature selection. Note that from Fig. 1 and Table 1 we see that the number of training images has a bigger impact than the sampling density of the extracted features: *e.g.* lowering from 100 to 50 the number of training images brings an 8% decrease in classification rates for NBSR_{l_1}, while reducing the feature density from 100 to 50 causes just a 2% drop. Each training image brings usually something new, while increased sampling over an image is likely to increase the redundancy of the features.

Feature density and asymmetry. What is better: higher sampling density in training or in query images? From Table 1 we see that for all the NB variants there is a higher drop in performance when we uniformly subsample in the query image than when we subsample in the training images. We achieve a 100 fold speedup by extracting a single NIMBLE feature either in each training image (causing a drop of 9% up to 12% in performance, depending on the method) or in the query (24% up to 35%). While in training the sampling density is less sensitive, for the query image the subsampling in fact removes important discriminant information for classification. For high feature densities NBSR_{l_1} and NB$INN$ are our best choices. In very low sampling densities ($\leq$ 10 per image), in either train or query or both, NBCR_{l_2} gets on par or better than NBSR_{l_1} and the other methods – promoting sparsity is less beneficial than using the whole data. It is worth mentioning that in lower densities NBNN_5 consistently outperforms NBNN_1 by stabilizing the assignments. While not tried, adjusting the neighborhood size to the pool size is expected to improve the results further.

Approximations. A consistent speed up can be obtained by using the structure of the data to drive the NN search. This is especially true when the feature dimensionality is low. Since NIMBLE features are 500 dimensional, we need to use approximated nearest neighbors. We control the chance of accurate neighbor retrieval to 90%, thus the drop in performance is marginal. Note that in our case, we achieve a maximal speedup of 5$\times$. The larger the training pool, the bigger the speedup. Also, the representations can be computed approximatively using a local neighborhood as retrieved using ANN.

State-of-the-art. Using NIMBLE features instead of SIFT features in a standard setup brings up to two orders of magnitude speed up in a straightforward implementation. This allows us to provide results for large datasets such as PASCAL VOC 2007 (see Table 3) where NBSR_{l_1} improves over NB$NN$ adapted for this task. For learning-free methods we show that representations such as CR_{l_2}, SR_{l_1} or INN are more suitable for high performance than the standard NN under the standard Naive Bayes Image Classification formulation. The methods are validated on Scene-15, Caltech-101, and PASCAL VOC 2007. NB$INN$ shows improvements over NBSR_{l_1} of 0.5% on Scene-15 and Caltech-101 (see Table 2) and of 1.4% on PASCAL VOC 2007 (see Table 3). Moreover, INN while being on par with SR_{l_2} it is much faster [9].

While we do not always outperform the state-of-the-art, it is surprising how close we get with our NB variants without any learning stage!

Best practice. For best performance, we suggest the use of NIMBLE features, high sampling densities in the query image (100 features per image), as many different training images as possible (not necessarily highly sampled), ANN for fast query feature neighborhood retrieval and more powerful sparse representation such as SR_{l_1} or INN. When the labeled pool is small (tens of images per class) and feature dimensionality is large (> 200), CR_{l_2} is a good option.

A good tradeoff between speed and performance is given by the following combination: NIMBLE features, low sampling densities (10 per image), ANN, and NBNN_5/NBINN for very small training pool of samples, NBCR_{l_2} for a small pool, or NBSR_{l_1}/NBINN for large pools.

6 Conclusions

In this work we have studied the use of sparse representations in a learning-free, parameter-free setup given by the Naive Bayes Classifier formulation. In particular, the NBSR_{l_1} and NB$INN$ give substantial improvements over the standard NB$NN$ approach which has the NN (hard assignment) as basis. Moreover, we have studied asymmetric schemes and the impact of the approximated nearest neighbors on the performance of the NB variants. Combined with recently introduced NIMBLE and G-KDES features, the NBSR_{l_1} and NB$INN$ achieves state-of-the-art results for learning-free methods in all the considered benchmarks. Moreover, on Caltech-101, we establish a new state-of-the-art for single descriptor based methods. On PASCAL VOC 2007, we get close to the best entry of the challenge, a learned complex approach. Naive Bayes Classification is still promising. Further directions are to better subsample by feature selection and to explore kernelized versions of the NB variants, learning the priors, and, thus, further improving over our basic methods.

Acknowledgement. This work was partly funded by the Flemish IWT/SBO project ALAMIRE and the ERC research grant COGNIMUND.

References

1. Boiman, O., Shechtman, E., Irani, M.: In defense of nearest-neighbor based image classification. In: CVPR, IEEE Computer Society (2008)
2. Behmo, R., Marcombes, P., Dalalyan, A.S., Prinet, V.: Towards optimal naive bayes nearest neighbor. In: ECCV (4). (2010) 171–184
3. Wang, Z., Hu, Y., Chia, L.T.: Image-to-class distance metric learning for image classification. In: ECCV (1). (2010) 706–719
4. Tuytelaars, T., Fritz, M., Saenko, K., Darrell, T.: The NBNN kernel. In: ICCV. (2011)
5. Yang, J., Yu, K., Gong, Y., Huang, T.S.: Linear spatial pyramid matching using sparse coding for image classification. In: CVPR, IEEE (2009) 1794–1801
6. Wright, J., Yang, A.Y., Ganesh, A., Sastry, S., Ma, Y.: Robust face recognition via sparse representation. PAMI **31** (2009)
7. Timofte, R., Van Gool, L.: Sparse representation based projections. In: BMVC. (2011)
8. Roweis, S., Saul, L.: Nonlinear dimensionality reduction by locally linear embedding. In: IEEE ICCV. Volume 290. (2001) 2323–2326
9. Timofte, R., Van Gool, L.: Iterative nearest neighbors for classification and dimensionality reduction. In: CVPR. (2012)
10. Zhang, L., Yang, M., Feng, X.: Sparse representation or collaborative representation: Which helps face recognition? In: ICCV. (2011)
11. Bo, L., Ren, X., Fox, D.: Kernel descriptors for visual recognition. In: Advances in Neural Information Processing Systems. (2010)
12. Kanan, C., Cottrell, G.W.: Robust classification of objects, faces, and flowers using natural image statistics. In: CVPR. (2010) 2472–2479
13. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. In: VISAPP (1). (2009) 331–340
14. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., Gong, Y.: Locality-constrained linear coding for image classification. In: CVPR. (2010) 3360–3367
15. Lee, H., Battle, A., Raina, R., Ng, A.Y.: Efficient sparse coding algorithms. In Schölkopf, B., Platt, J.C., Hoffman, T., eds.: NIPS, MIT Press (2006) 801–808
16. Timofte, R., Van Gool, L.: Weighted collaborative representation and classification of images. In: ICPR. (2012)
17. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60** (2004) 91–110
18. McCann, S., Lowe, D.: Local naive bayes nearest neighbor for image classification. In: CVPR. (2012)
19. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR (2). (2006)
20. Fei-Fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE Transactions on Pattern Analysis and Machine Intelligence **28(4)** (2006) 594–611
21. Feng, J., Ni, B., Tian, Q., Yan, S.: Geometric lp-norm feature pooling for image classification. In: CVPR, IEEE (2011) 2697–2704
22. Gao, S., Tsang, I.W.H., Chia, L.T., Zhao, P.: Local features are not lonely - laplacian sparse coding for image classification. In: CVPR. (2010) 3555–3561
23. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. (http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html)