Iterative Nearest Neighbors for Classification and Dimensionality Reduction

Radu Timofte and Luc Van Gool
ESAT-PSI-VISICS / IBBT
Katholieke Universiteit Leuven, Belgium
{firstname.lastname}@esat.kuleuven.be

Abstract

Representative data in terms of a set of selected samples is of interest for various machine learning applications, e.g. dimensionality reduction and classification. The best-known techniques probably still are k-Nearest Neighbors (kNN) and its variants. Recently, richer representations have become popular. Examples are methods based on l_1-regularized least squares (Sparse Representation (SR)) or l_2-regularized least squares (Collaborative Representation (CR)), or on l_1-constrained least squares (Local Linear Embedding (LLE)). We propose Iterative Nearest Neighbors (INN). This is a novel sparse representation that combines the power of SR and LLE with the computational simplicity of kNN. We test our method in terms of dimensionality reduction and classification, using standard benchmarks such as faces (AR), traffic signs (GTSRB), and PASCAL VOC 2007. INN performs better than NN and comparable with CR and SR, while being orders of magnitude faster than the latter.

1. Introduction

The point of departure for dimensionality reduction techniques often is a property that ought to be preserved or enhanced under the reducing projection. Examples of linear techniques are Principal Component Analysis (PCA)[22] (enhances the variance), Locality Preserving Projections (LPP)[9] (conserves the local similarities), Sparse Representation Linear Projections[15] (embeds the sparse representation), Linear Discriminant Analysis (LDA)[7, 13] (enhances the inter-class variance in comparison to the intra-class variance). Examples of non-linear techniques are Locally Linear Embedding (LLE)[14] (preserves the local neighborhood), Laplacian Embedding (LE)[2] (preserves the distances to neighbors), or Sparse Representation Embedding (SRE)[15] (preserves the sparse representation).

In the context of classification, we define the criteria / properties to match a given query sample against the known, labeled samples or the learned class model. The simplest such decision is directly picking the label of the nearest neighbor as the class label for the query. 'Nearest' is defined on the basis of some similarity, distance, or metric. The Sparse Representation-based Classifier (SRC) proved to yield state-of-the-art performance in face recognition [18]. This classifier starts from a Sparse Representation (SR) in the l_1-regularized least squares sense. It then decides based on the class labels of the samples that contribute to the representation of the query sample. The Collaborative Representation Classifier (CRC) [21] starts instead from the l_2-regularized least squares solution.

Of course, apart from performance, there also is the computational efficiency to consider. Whereas SR and CR may excel in terms of the former, these methods are also computationally demanding. On the other hand, the performance of kNN may be less impressive, but in comparison it is very fast.

We aim at combining the high speed of kNN with the performance of SR. Our proposal, coined the Iterative Nearest Neighbors representation, is inspired by feedback loop methods. First, instead of searching for the weights in the representation like most current l_1 solvers, INN instead searches for the nonzero terms. Moreover, the selection of every additional known sample is supposed to compensate for the errors introduced by the previous selections and thus further reduce the residual error between the query sample and its reconstruction. This leads to an iterative procedure. Starting from the query sample, INN first gets its nearest neighbor. In the next iterations, it each time picks the known sample best suited to reduce the error that remains between the query sample and its reconstruction on the basis of the previously selected, known samples. This process is repeated until either a maximum number of sample terms is reached, the residual has fallen below a threshold, or the sum of the remaining weights in a potentially infinite representation is too small compared to the sum of the weights already assigned to samples in the representation. In order to transform the selection of each new term into a nearest neighbor decision, we adapt the query at each step by adding the weighted residual introduced by the previous

selection. This weight is the regularization parameter of the method and is fixed and taken to lie in the (0,1) interval. As will become clear, the sum of the weights in the representation is guaranteed to have a limited value. Moreover, after a limited number of iterations, the remaining terms can be discarded, once the application's error tolerance has been reached.

INN follows a simple procedural construction, with a complexity equal to the number of non-zeroes in the representation $\times$ the time complexity of the nearest neighbor search. Experimentally, we found the performance to be robust for a wide range of values for the parameter regulating the importance of the residual introduced by each new picked nearest neighbors. It seems a fairly stable parameter, similar to the Lagrangian parameter used in l_1/l_2-regularized least squares solvers. Nevertheless, tuning the parameter for specific data usually still brings some improvement (as with l_1/l_2-solvers).

The remainder of the paper is structured as follows. First, we introduce the Iterative Nearest Neighbors method in Section 2. Then, we provide an analysis of its time complexity, bounds and approximations in Section 3. In Section 4 we derive our INN-based variant of the SRLP dimensionality reduction method. Section 5 introduces our INN-based Classifier and derives INN-based variants for the Naive Bayes classifier and the Sparse Coding Spatial Pyramid Matching method. The conclusions are drawn in Section 7.

2. Iterative Nearest Neighbors (INN)

The Iterative Nearest Neighbors (INN) algorithm follows a constructive definition. For a finite set $\mathbf{X}$ of finite training samples $\mathbf{x} \in \mathbb{R}^D$ and a new sample $\mathbf{q} \in \mathbb{R}^D$, that is to be approximated with the help of a series of working queries $\mathbf{q}'$, the first of which $\mathbf{q}' = \mathbf{q}$, the steps are as follows

1) Find the nearest neighbor in $\mathbf{X}$ ($NN(\mathbf{q}')$) for the current query $\mathbf{q}'$.
2) Update the query q' by adding the weighted residual. $\mathbf{q}' = \mathbf{q}' + \lambda(\mathbf{q}' - NN(\mathbf{q}'))$
3) Repeat the steps until the stopping criterion is satisfied.

The idea of continuously compensating for the residuals at each stage of the approximation is akin to regulators adding a regulatory value during each feedback loop up to some final convergent behavior of the system output. λ is the regulatory parameter of the system. It specifies the amount of the residual image to be used in the adapted query for the next iteration.

From the construction, we have the following formalism, where k is the iteration step and $\mathbf{s}_k = NN(\mathbf{q}_k)$:

$$\begin{array}{rcll} \mathbf{q}_k & = & \mathbf{q}, & k=1 \\ \mathbf{q}_k & = & \mathbf{q}_{k-1} + \lambda(\mathbf{q}_{k-1} - \mathbf{s}_{k-1}), & k>1 \\ & = & (1+\lambda)\mathbf{q}_{k-1} - \lambda\mathbf{s}_{k-1}, & k>1 \end{array} \quad (1)$$

From this we can reconstruct the original query as by recursive substitution:

$$\begin{array}{rcl} \mathbf{q} & = & \mathbf{q}_1 \\ & = & (\mathbf{q}_2 + \lambda\mathbf{s}_1)/(1+\lambda) \\ & = & ((\mathbf{q}_3 + \lambda\mathbf{s}_2)/(1+\lambda) + \lambda\mathbf{s}_1)/(1+\lambda) \\ & = & \frac{1}{(1+\lambda)^2}\mathbf{q}_3 + \frac{\lambda}{(1+\lambda)^2}\mathbf{s}_2 + \frac{\lambda}{(1+\lambda)^1}\mathbf{s}_1 \\ & \ldots & \\ & = & \frac{1}{(1+\lambda)^{k-1}}\mathbf{q}_k + \sum_{i=1}^{k-1}\frac{\lambda}{(1+\lambda)^i}\mathbf{s}_i \end{array} \quad (2)$$

So, in the reconstruction of the original query, each selected sample, $\mathbf{s}_i$, has an a priori known (imposed by the algorithm) weight, $w_i \in (0,1)$, depending on $\lambda \in (0,1)$ and iteration, i:

$$w_i = \frac{\lambda}{(1+\lambda)^i} \quad (3)$$

Now, we can provide a more detailed algorithm, where as stopping criterion we use the maximum number of iterations K and the reconstruction error r. Also, Figure 1 shows how the INN algorithm constructs the sparse representation for a query sample, iteration by iteration.

Input:
$\mathbf{q} \in \mathbb{R}^D$ - the query, X - set of finite samples $\in \mathbb{R}^D$,
$\lambda \in (0,1)$ - the parameter, $\epsilon \in \mathbb{R}_+$ - error tolerance,
$K \in \mathbb{N}_+$ - maximum number of terms

Output:
$\{\mathbf{s}\}_1^k$ - the used samples, $\{w\}_1^k$ - weights

The INN Algorithm:

0) Initialize: $k=0$, $\mathbf{q}' = \mathbf{q}$, $\hat{\mathbf{q}} = 0$, $w_0 = \lambda$
1) NN search in $\mathbf{X}$: $k = k+1$, $\mathbf{s}_k = NN(\mathbf{q}')$, $w_k = \frac{w_{k-1}}{1+\lambda}$
2) Approximation and error update: $\hat{\mathbf{q}} = \hat{\mathbf{q}} + w_k\mathbf{s}_k$, $r = \|\mathbf{q} - \hat{\mathbf{q}}\|_2$
3) Adapt the query: $\mathbf{q}' = \mathbf{q}' + \lambda(\mathbf{q}' - \mathbf{s}_k)$
4) If $k < K$ and $r > \epsilon$ goto **1)**.

3. Theoretical Analysis/Bounds

The algorithmic recursive procedure of INN brings us to the following formulation for obtaining the sparse representation terms:

$$\arg\min_{\{\mathbf{s}\}_{i=1}^{\infty}} \{\mathbf{q} - \sum_{i=1}^{\infty}\frac{\lambda}{(1+\lambda)^i}\mathbf{s}_i\} \quad (4)$$

We notice that the maximum sum of the weights is

$$\lim_{K\to\infty}\{\sum_{i=1}^{K}\frac{\lambda}{(1+\lambda)^i}\} = 1 \quad (5)$$

for any $\lambda \in (0,1)$, and regardless the selection of $\mathbf{s}_i \in X$

$$\sum_{i=1}^{\infty}|\frac{\lambda}{(1+\lambda)^i}\mathbf{s}_i| \le \max_{\mathbf{x}\in X}|\mathbf{x}|\sum_{i=1}^{\infty}\frac{\lambda}{(1+\lambda)^i} = \max_{\mathbf{x}\in X}|\mathbf{x}| < \infty \quad (6)$$

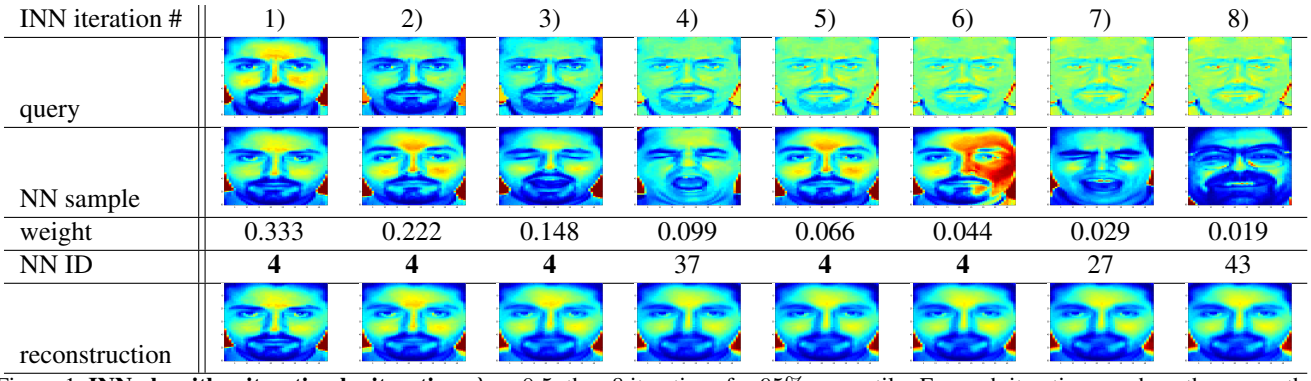


Figure 1. **INN algorithm iteration by iteration.** $\lambda = 0.5$, thus 8 iterations for 95% percentile. For each iteration we show the query, the selected NN sample from the training set and its ID, the assigned weight, and the reconstruction up to that iteration. The INNC cumulates a 0.813 recognition confidence for the ID 4.

therefore, the obtained representation is absolute convergent and convergent to a finite number, not necessarily $\mathbf{q}$.

For the number K of terms necessary to sum up to a β-fraction of the total sum of the weights, we find:

$$\beta = (\sum_{i=1}^{K} \frac{\lambda}{(1+\lambda)^i})/1 = 1 - (1+\lambda)^{-K} \tag{7}$$

from where

$$K = \lceil -\frac{\log(1-\beta)}{\log(1+\lambda)} \rceil \tag{8}$$

Thus, we can safely search only for the first K terms in the query expansion to obtain the most important weights and thus, a large drop in computational cost is achieved.

Structured Representations The INN solution as proposed here is an approximation of the full problem (4). The sparsity is imposed implicitly by the geometrical progression of the weights.

The sparse representation computed here is a particular case of what could be a large class of structured representations where the weights are imposed. For that class, the optimization is done over the known samples to satisfy the imposed weights, thus revealing an imposed structure in the data. This is opposite to many of the sparse representation formulations and solvers, which usually optimize over the weights and consider the samples fixed. Imposing the weights may be counter-intuitive, but as the experiments will show, brings important advantages.

From a different perspective, the INN algorithm is choosing, at each iteration, a sample and updates its weight with a decaying known value controlled by λ and the iteration number. As we saw, the weights assigned to the selected samples are combinations of a discrete, known set with size equal to the iteration number.

If we impose the number of iterations (maximum number of non-zeroes) and the approximation tolerance as a percentage of the completely expanded sum of the weights, then we directly know the regulatory parameter λ using the relation (8). In our experiments the percentile is set to 95% ($\beta = 0.95$), so that the remaining 5% corresponds to the measurement noise, i.e. the representation tolerance. The resulting representation converges (not necessarily to the query sample) in the limit. In practice, the iterations can be aborted as soon as the representation residual or the sum of the remaining weights is in check (less than the data noise).

By imposing the positive weights in eq. (3), and thus their maximum sum, as well as the number of terms, and thus the maximum number of non-zeroes, INN acts like an l_1-constrained recurrent residual decomposition. The choice of distance for the determination of the nearest neighbor is an open question. The choices considered here are l_1 and l_2. The first assigns an equal importance to all feature vector elements, the second more strongly penalizes the big differences.

4. Dimensionality Reduction

Recently, a Sparse Representation based Linear Projective (SRLP) method has been proposed [15], as a variant of Locality Preserving Projections (LPP) [9], where the weights are replaced by the absolute coefficients of the l_1-regularized least squares solution. Our INN based Linear Projection (INNLP) is motivated by the reported improvement in classification performance of SRLP over the original LPP and Linear Discriminant Analysis (LDA) projections. In fact the INNLP framework is identical to that of SRLP, only the weights are now obtained by solving the INN representation and not an l_1-regularized least squares problem. Here we closely follow the exposition from [15] as used for SRLP, for introducing our INNLP variant.

SRLP and our INNLP variant aim at preserving the weights from the sparse representation as similarities in the linear embedding space. There are three steps: adjacency graph construction, weight computation, and eigenmapping. The obtained weighted graph is assumed to correspond to a symmetric matrix.

INNLP (and SRLP) replaces the graph representation with a matrix of weights $\mathbf{W}_{N\times N}$ coming from the INN sparse representations.

For each training point $\mathbf{x}_i$ we compute the sparse representation given by the INN algorithm. Let $\mathbf{w}_i \in \mathbb{R}^N$ be these weights, where the non-zeroes are given directly by INN, and $\mathbf{w}_{ij}$ the contribution of $\mathbf{x}_j$ to the sparse representation.

In a supervised sparse representation scenario based on labels, we restrict the INN representation to be picked from the samples sharing the same class label.

The INNLP weighted symmetric adjacency matrix is then defined as

$$\mathbf{W}_{N\times N} = \max\{[\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_N], [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_N]^T\} \tag{9}$$

The third step, common to the original LPP and SRLP, is solving a generalized eigenvector problem:

$$\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{a} = \lambda\mathbf{X}\mathbf{D}\mathbf{X}^T\mathbf{a} \tag{10}$$

where $\mathbf{D}$ is a diagonal matrix, $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ji}$. The Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$. The matrix $\mathbf{X}$ has the training samples, $\mathbf{x}_i$, as columns.

If $\mathbf{a}_1, \cdots, \mathbf{a}_P$ are the solutions of (10), sorted by their corresponding eigenvalues, $\lambda_1 < \cdots < \lambda_P$, then the embedding is:

$$\mathbf{x}_i \rightarrow \mathbf{y}_i = \mathbf{R}^T\mathbf{x}_i, \mathbf{R} = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_P], \mathbf{R} \in \mathbb{R}^{M\times P} \tag{11}$$

Similarly, one can replace the sparse representation from the non-linear Sparse Representation-based Embedding [15] or other graph based methods to derive INN-based variants.

5. Classification

5.1. INN-based Naive Bayes Image Classification

In [4], the powerful Naive Bayes Nearest Neighbor (NBNN) image classifier has been proposed. To a certain extent, it is a learning-free and parameter-free method, it avoids discretization, and it generalizes well using 'image-to-class' instead of 'image-to-image' comparisons.

We derive two Naive Bayes variants: i) NBINN - INN-based Naive Bayes classifier, and, ii) NBSR$_{l_1}$ using the sparse representations obtained from the l_1-regularized least square formulation.

The Naive Bayes classifier [4] uses strong independence assumptions. For a query image, represented as a set $\mathbf{Q}$ of independently sampled features, $\mathbf{q}_i \in \mathbf{Q}$, from a class-specific feature distribution $p(\mathbf{q}|c)$, and with assumed uniform priors, $p(c)$, the classification is given by

$$\begin{aligned}\hat{c} &= \arg\max_{c\in\mathbf{C}} p(c|\mathbf{Q}) = \arg\max_{c\in\mathbf{Q}}\{\textstyle\prod_{\mathbf{q}\in\mathbf{Q}} p(\mathbf{Q}|c)\} \\ &= \arg\max_{c\in\mathbf{C}}\{\textstyle\sum_{\mathbf{q}\in\mathbf{Q}} \log p(\mathbf{q}|c)\}\end{aligned} \tag{12}$$

For a given sample $\mathbf{q}$ and a matrix $\mathbf{X}$ with M labeled samples stacked column-wise, $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_M]$, NBSR$_{l_1}$ optimizes over $\mathbf{w}$:

$$\min_{\mathbf{w}} \|\mathbf{q} - \mathbf{X}\mathbf{w}\|_2 + \lambda\|\mathbf{w}\|_1 \tag{13}$$

Then, the absolute values of the weights are taken as a measure of their importance for the query $\mathbf{q}$ and further l_1-normalized to sum to 1:

$$\hat{\mathbf{w}}^q = |\mathbf{w}|/\|\mathbf{w}\|_1 \tag{14}$$

NBINN already has nonnegative weights from its INN representation and their sum is already bounded by 1. Now the likeliness of $\mathbf{q}$ to belong to class c is given by:

$$d_q^c = \sum_{\mathbf{x}\in\mathbf{X}_c} \hat{\mathbf{w}}_{\mathbf{x}}^q \tag{15}$$

where $\mathbf{X}_c$ is the submatrix of $\mathbf{X}$ corresponding to class c.

The decision for the NBSR$_{l_1}$ and the NBINN variant, is taken using:

$$\hat{c} = \arg\max_{c\in\mathbf{C}} \sum_{\mathbf{q}\in\mathbf{Q}} d_q^c \tag{16}$$

and gives the class which cumulates the largest importance/likeliness in the sparse representations of the query image $\mathbf{Q}$.

5.2. INN-based Classification

In [18], the Sparse Representation-based Classifier has been proposed in the context of face recognition. The original formulation uses the residuals for decision making and explicitly deals with noise. In [15] the decision is taken directly on the basis of the absolute values of the nonzero coefficients, as importance indicators for each query sample.

Here we closely follow the exposition from [15], and for our INN-based Classifier (INNC), we replace the sparse representation with our proposed INN representation.

For an unknown query sample $\mathbf{q} \in \mathbb{R}^D$, we obtain the INN representation over the training set where $\mathbf{v} \in \mathbb{R}^N$ are the sparse representation weights and the decision is taken using

$$\hat{c} = \arg\max_{c\in\mathbf{C}} \sum_{t_i=c} \mathbf{v}_i \tag{17}$$

where $t_i \in \mathbf{C}$ is the class label and $\mathbf{v}_i$ is the weight corresponding to the i-th sample, $\mathbf{x}_i$. The class $\hat{c}$ has the largest impact in the representation of the query sample $\mathbf{q}$. This is the INN-based Classifier decision as used in our experiments.

5.3. INN Spatial Pyramid Matching

In [20], the Sparse Coding Spatial Pyramid Matching method has been proposed for image classification. The sparse coding, seen as a 'soft assignment' over the learned dictionary, uses the l_1-regularized least squares formulation.

Replacing this formulation with our INN representation is straightforward. The reader is referred to the original work [20] for the details and the implementation. In our experiments, this variant is named INNSPM.

6. Experimental Results

In this section we describe the experimental setups, the results obtained, and provide a discussion. We aim at providing a thorough evaluation of our proposed INN representations and our INN based methods for specific tasks: dimensionality reduction with INNLP, classification with INNC, Naive Bayes image classification (in a retrieval setting) with NBINN, and moreover we modified Sparse Coding Spatial Pyramid Matching for image classification.

6.1. Regularization parameters

To evaluate how the regularization parameter affects the performance of our INN representation we use the INNC classifier and the AR benchmark as used in [21], i.e. as a subset (only expression and illumination changes) of the original face dataset. We have 100 individuals, 7 images for each for training from Session 1, and 7 for each for testing from Session 2. The images are cropped to 60x43 pixels. We use Regularized LDA to obtain projected features. Here, the RLDA parameter is set to 0.001. We work with 30-dimensional embeddings and 99-dimensional embeddings to capture the behavior of the classifiers in low and higher dimensional spaces. The results are depicted in Figure 2.

Usually a measurement noise level or a representation tolerance is assumed a priori, which translates for our INN into a percentage of the sum of weights for the full decomposition. We fix the percentage to 95% for all the experiments involving INN. The number of INN iterations is obtained using eq. (8). The only parameter of INN is λ.

For INNC, the lower λ, the better the recognition rate but also the slower the method. We have a well-behaved performance with respect to the INN parameter. Note that with $\lambda = 0.05$ (which corresponds to 62 iterations using (8) and target 95%), the performance gain is negligible. When λ gets closer to 1, the INNC performance drops to the one achieved using a Nearest Neighbor (NN) classifier. If $\lambda = 0.25$, the INNC performance is still very close to the best achievable recognition rate, but the number of iterations is equal to 14, thus only requiring 14 times the time for a NN search.

In the same setting, we evaluate the impact of the parameters on the Collaborative Representation Classifier (CRC) [21] and Sparse Representation Classifier [18] in the form from [21], using residuals. The classifier performances are directly related to the power of the representations used.

Note that the CRC classifier uses l_2-normalized features as input. All the features in our experiments are l_2-normalized prior to applying the classifiers. CRC computes offline a projection matrix which involves computing the inverse of a matrix. A small value λ is added to the diagonal to stabilize the solution; this is the regularization parameter for CRC.

For the SRC classifier we used various l_1-regularized least squares solvers: L1LS [11], Homotopy [5, 1], and Feature Sign [12]. The reader is referred to [19] for a comparison study of l_1-solvers. The l_1-solvers tackle the following problem:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \|\mathbf{q} - \mathbf{X}\mathbf{w}\|_2 + \lambda\|\mathbf{w}\|_1 \qquad (18)$$

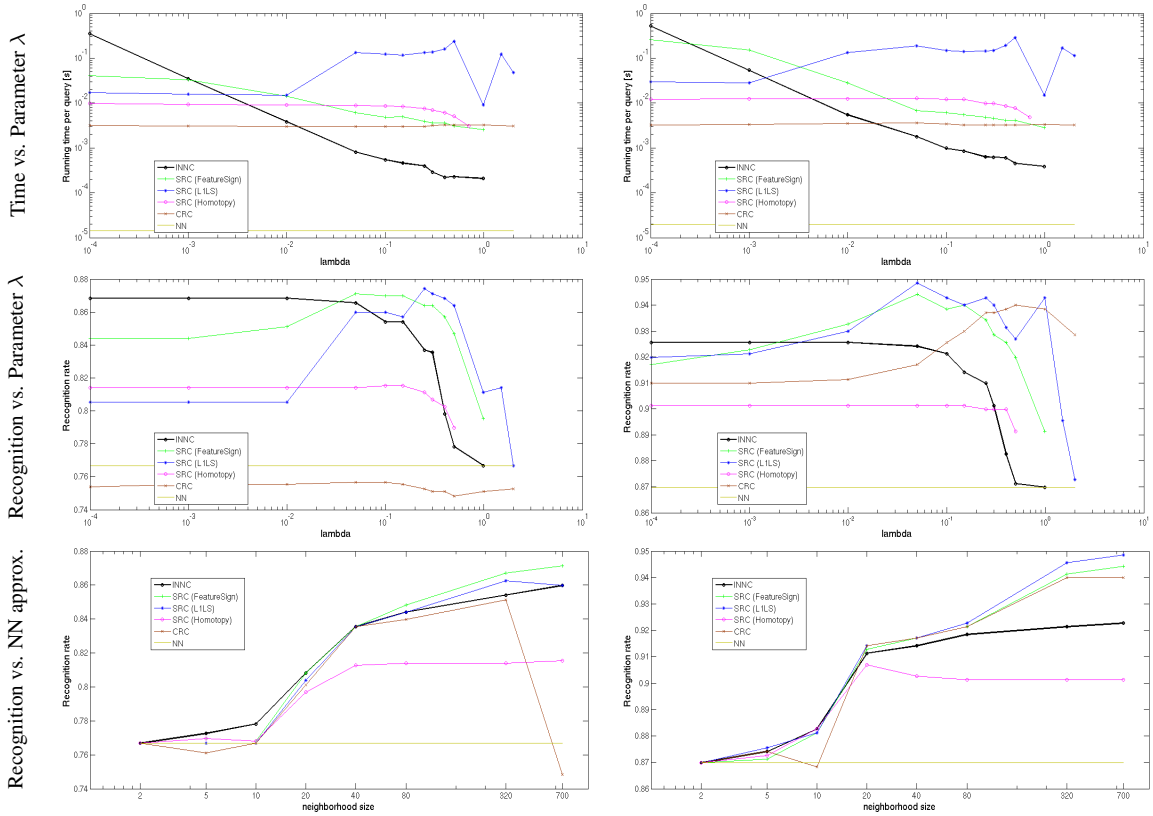
We use a tolerance of 0.05, for the FeatureSign, L1LS, and Homotopy solvers, as in [18, 15, 19]. λ remains the regulatory parameter between sparsity and reconstruction error for these solvers. The L1LS and FeatureSign based SRC are the top performers achieving their best recognition rate for $\lambda \in [0.01, 1]$. SRC (FeatureSign) heavily benefits if sparsity is enforced (larger λ), and in some cases is able to come close in terms of running time to CRC. The running time for CRC in Figure 2 does not contain the 'offline' time for computing the projection matrix, which is considerably high for large training sets.

For all the classifiers studied here (except NN), setting λ is a critical step for achieving the best performance. Nevertheless, most of these classifiers have 'safety intervals' for λ. INNC gets close to its best performance for $\lambda \in (0, 0.2]$, and SRC with Homotopy for $(0, 0.1]$, with L1LS for $(0.01, 1]$, and with FeatureSign for $(0.01, 0.4]$.

According to the experiments, our proposed INNC classifier can achieve on par performance with the best SRC (L1LS) and SRC (FeatureSign) classifiers, but being one (FeatureSign) up to 3 orders (L1LS) of magnitude faster. SRC (Homotopy) proved to be slower and poorer in performance than the FeatureSign algorithm. The CRC 'online' time does not depend on λ, since there λ is used in the 'offline' processing of the projection matrix. Still, our INN-based classifier can be more than 10 times faster than CRC, mainly due to the fact that INNC uses the weights and not the residuals (as CRC does) for taking the decision.

When compared to NN, as expected, all the other methods are slower. The number of iterations of INNC stands in a direct linear relation with the number of NN searches that need to be performed, and, hence, with how many times slower the INNC is than NN.

In practice for speed gain, the representations are computed in a neighborhood of the query sample. This case is



a) 30 dimensional LDA projection *b)* 99 dimensional LDA projection

Figure 2. Parameter influence on performance on the AR dataset, with RLDA projections of dimensionality 30 *a)*, and 99 *b)*. INNC has a well-behaved performance w.r.t. the parameter, regardless of embedding dimensionality. The lower λ is, the better the performance of INNC. For CRC we did not include the time for computing the projection matrix.

depicted in Figure 2. INNC is on par with the best SRC for low neighborhood sizes (≤ 80).

6.2. Dimensionality Reduction

For evaluating the performance of our INNLP variant against the original SRLP we use the AR faces benchmark and GTSRB traffic sign benchmark. For AR we keep the same settings. GTSRB has 43 classes with 39209 training images and 12630 test images. All the annotations are cropped and normalized to 24×24 pixels patches. The original features are the gray scale pixel values, l_2-normalized. The linear projection matrices are learned over these features in a supervised setting, on the training data. The l_2-normalized projected features are the input for different classifiers, such as the NN, SRC, CRC and Support Vector Machines (SVM) with Linear (L), Intersection (IK), Polynomial (POLY) and Radial Basis Function (RBF) kernels.

Table 1 depicts the face recognition results using our INNLP method vs. SRLP, Eigenfaces and LDA. We set the regularization parameter to be 0.001 in order to avoid the singularity in all the projective methods. INNLP uses $\lambda = 0.05$ for computing the INN representations and SRLP uses FeatureSign with $\lambda = 0.05$. INNLP projections prove to be as good as the SRLP$_{FeSg}$ projections for all the classifiers considered, but the projection learning is more than 10 times faster. At equal embedding dimensionality, LDA, SRLP, and INNLP are comparable or better than Eigenfaces. This was to be expected since Eigenfaces are obtained in an unsupervised fashion.

In Table 2 we compare INNLP with Eigenfaces, SRLP and LDA. Again, INNLP is on par with SRLP$_{FeSg}$ using the FeatureSign solver, but learns faster. It learns more than ten times faster than SRLP$_{Hmtp}$ using the Homotopy solver. Note that the LDA projection better fits the NN classifier than the sparse representation based projections.

6.3. Classification

We evaluate the NBINN, NBSR$_{l_1}$ and INNSPM methods against the standard NBNN, the recently proposed NBSR$_{l_1}$, and the original ScSPM. For this purpose we are using the Scene-15 dataset and PASCAL VOC 2007 benchmark. We keep the same setup for feature extraction as in [10]. The NIMBLE features are extracted using 100 (eye) fixations per image, and are PCA-projected into a 500-dimensional space. For NBSR$_{l_1}$ we use the FeatureSign solver with $\lambda = 0.3$ for the sake of speed. NBINN uses INN with $\lambda = 0.1$,

Table 1. The face recognition results [%] of different methods on AR database using different linear projections.

	Dim	5	10	30	54	99	120	300
eigenfaces	NN	23.5	43.4	59.1	68.1	69.8	70.4	71.4
	LSVM[21]				69.4		74.7	75.4
	CRC	06.3	19.5	64.2	80.3	89.3	90.1	**93.8**
	SRC(Hmtp)	23.3	**48.6**	69.7	75.1	77.4	78.1	79.8
	SRC(FeSg)	23.3	46.1	**70.8**	76.7	80.4	81.0	82.7
	SRC(L1LS)	06.2	19.6	64.7	**81.0**	**90.0**	**91.4**	**93.4**
	INNC	**24.1**	**48.4**	68.8	74.1	77.1	77.7	79.4
	Dim	5	10	30	54	99	120	300
LDA	NN	**37.2**	58.8	76.7	81.8	87.0		
	CRC	09.6	26.0	75.5	90.7	91.0		
	SRC(FeSg)	34.5	56.8	**87.1**	**92.1**	**94.4**		
	SRC(L1LS)	20.0	47.9	86.0	**92.4**	**94.9**		
	INNC	**37.5**	**60.8**	**86.6**	88.8	92.6		
	Dim	5	10	30	54	99	120	300
SRLP_{FeSg}	NN	37.8	59.8	77.3	82.8	86.0	86.4	86.7
	CRC	09.9	27.3	77.0	90.0	91.8	92.4	88.4
	SRC(FeSg)	34.9	61.4	**86.3**	**92.0**	**94.0**	**94.3**	**94.4**
	SRC(L1LS)	18.7	40.9	81.8	91.3	93.1	93.7	**94.4**
	INNC	**39.4**	**62.3**	85.3	88.6	92.4	93.0	93.1
	Dim	5	10	30	54	99	120	300
INNLP	NN	**37.3**	59.7	77.4	82.7	86.0	86.1	86.9
	CRC	09.7	27.6	76.3	89.6	91.7	92.6	88.0
	SRC(FeSg)	34.9	60.0	**86.1**	**92.3**	**94.1**	**94.0**	**94.1**
	SRC(L1LS)	18.3	39.8	81.4	90.7	93.0	**93.7**	**94.4**
	INNC	**37.3**	**60.8**	85.1	89.0	92.7	93.1	93.0

Table 2. The traffic sign classification accuracy[%] on GTSRB benchmark. We use 300 eigenfaces, 42-dimensional LDA projections, 99-SRLP, 99-INNLP.

classifier	eigenf.	SRLP_{FeSg}	INNLP	SRLP_{Hmtp}	LDA
NN	66.05	89.54	89.35	89.27	91.84
INNC	77.70	**93.64**	**93.61**	**93.61**	**93.64**
SRC(Hmtp)	76.53	93.01	93.06	92.45	92.39
SRC(FgSg)	**85.31**	**93.94**	**93.74**	92.93	92.91
SRC(L1LS)	74.78	79.34	79.41	93.13	93.01
LSVM		87.87		87.38	87.00
IKSVM		89.51		89.66	86.30
POLYSVM		92.76		92.51	92.49
RBFSVM		92.43		92.28	92.46

Table 3. Performance Comparison on Scene-15

Learned?	Train/test split Method	100/100 avg CR
yes	ScSPM+SIFT [20]	80.28 ± 0.93
yes	EMK+KDES [3]	87.5
yes	LScSPM+SIFT [8]	**89.75±0.5**
yes	NBNN&BoF kernels+SIFT [16]	85 ± 4
yes	NBNN-f_2+SIFT [16]	75 ± 3
yes	INNSPM+SIFT	81.4 ± 1
no	NBNN+NIMBLE	74.2 ± 1
no	NBSR_{l_1}+NIMBLE	77.75 ± 1
no	NBINN+NIMBLE	**78.23±1**

and INNSPM with $\lambda = 0.1$.

For Scene-15 we report the results in Table 3. The proposed NBINN is slightly better than NBSR_{l_1} and both improve over NBNN. Moreover, the results are comparable with the results obtained using spatial pyramidal matching and SIFT. Significant is also the $10\times$ speedup achieved by NBINN over NBSR_{l_1}, which makes it comparable in running time to the NBNN approach. Similarly, the speedup of INNSPM over ScSPM is one order of magnitude, giving the method the same run-time as the LLC method. The reported performance of INNSPM is on par with or better than that of LLC and the original ScSPM, and better than the learning-free NB approaches.

For the PASCAL VOC 2007 dataset we report the results in Table 4. Here, NBINN outperforms the NBSR_{l_1} classifier both in speed and performance. The achieved mean average precision is only 6% below the best entry of the PASCAL VOC 2007 challenge. The NB classifiers we use here are based on a single type of features, the NIMBLE features. The performance achieved by INNSPM+SIFT is better than that of ScSPM+SIFT or LLC+HOGs, while the running time compares favorably to the LLC method.

6.4. Discussion

In this paper we propose a novel sparse representation, namely Iterative Nearest Neighbors. It combines the speed of the kNN approach with the performance of sparse decompositions obtained using the l_1-regularized least squares formulation.

Moreover, we presented: (i) our INN Linear Projection method for dimensionality reduction, shown to improve over the original SRLP, (ii) our Naive Bayes INN for image classification, shown to improve over the NBNN and NBSR_{l_1} classifiers on the PASCAL VOC 2007 benchmark and reaching state-of-the-art results, (iii) our INN based Classifier, shown to be faster and on par with the l_1-based SRC and the l_2-based CRC for face recognition (AR benchmark), and being on par on traffic signs (GTSRB benchmark) with the best SRC and FeatureSign solver, but order(s) of magnitude faster, and (iv) our INNSPM pipeline for image classification, showing improvement both in speed and performance over the original ScSPM [20] based on solving l_1-regularized least squares, and having comparable speed and better performance when compared with Locality-Constraint Linear Coding (LLC) [17], acknowledged to be one of the fastest and most strongly performing image classification frameworks, when run on PASCAL VOC 2007.

Moreover, INN has desirable properties, such as:

(i) steep convergence to a ϵ solution,

(ii) uses nearest neighbors as main computational step,

(iii) has only one regulatory parameter, similar to the Lagrangian parameter in regularized least squares techniques,

(iv) direct formulas for the nonzero weights and for the number of significant nonzero terms.

On the downside, INN does not scale well with high-dimensional data, as seen in Figure 2.

7. Conclusions

In this paper we have proposed a novel sparse representation, namely Iterative Nearest Neighbors (INN). INN combines the power of l_1/l_2-regularized least squares representations with the simplicity and speed of kNN. The definition

Table 4. Image classification results on PASCAL VOC 2007 benchmark

object class	aero	bicyc	bird	boat	bottle	bus	car	cat	chair	cow	
Best of VOC'07[6]	77.5	63.6	56.1	71.9	33.1	60.6	78.0	58.8	53.5	42.6	
LLC [17]	74.8	65.2	50.7	70.9	28.7	68.8	78.5	61.7	54.3	48.6	
NBNN	70.7	59.3	40.3	47.4	23.0	57.4	74.3	50.7	42.2	32.9	
NBSR_{l_1}	67.9	63.6	47.0	50.5	22.4	57.9	75.3	56.0	47.0	34.6	
NBINN	69.8	63.8	48.5	61.9	26.6	58.9	74.8	52.9	51.6	36.0	
INNSPM	77.2	64.4	56.2	71.4	32.7	69.1	80.0	59.8	49.5	47.9	
object class	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	average
Best of VOC'07[6]	54.9	45.8	77.5	64.0	85.9	36.3	44.7	50.9	79.2	53.2	59.4
LLC [17]	51.8	44.1	76.6	66.9	83.5	30.8	44.6	53.4	78.2	53.5	59.3
NBNN	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}	37.9	41.6	74.8	62.5	81.7	21.4	28.1	43.5	73.1	46.2	51.7
NBINN	42.5	40.8	75.7	62.2	82.8	22.1	28.9	41.3	74.1	46.0	**53.1**
INNSPM	55.3	45.8	77.8	67.0	84.6	30.2	44.7	53.4	79.1	53.8	**60.0**

is constructive, as it compensates the residuals by iteratively picking the nearest neighbor to the perturbated input query. The impact of the residual is the controlling parameter for this procedure. It allows for the direct computation of the number of iterative steps (thus, the maximum number of neighbors) that is to be conducted assuming measurement noise. INN is well behaved, the smaller the parameter λ, the better the performance but the slower it gets. INN exhibits peformance levels that are on par with those of CR and SR for classification and dimensionality reduction tasks, while being orders of magnitude faster.

Acknowledgments. This work was partly supported by the European Commission FP7-231888-EUROPA project and the IWT/SBO ALAMIRE project.

References

[1] M. S. Asif. Primal dual pursuit: A homotopy based algorithm for the dantzig selector. *M.S. Thesis. Georgia Institute of Technology*, 2008. 5

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press, 2001. 1

[3] L. Bo, X. Ren, and D. Fox. Kernel descriptors for visual recognition. In *Advances in Neural Information Processing Systems*, December 2010. 7

[4] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*. IEEE Computer Society, 2008. 4

[5] D. L. Donoho and Y. Tsaig. Fast solution of l_1-norm minimization problems when the solution may be sparse. *IEEE Transactions on Information Theory*, 54(11):4789–4812, 2008. 5

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 8

[7] R. Fisher. The statistical utilization of multiple measurements. *Annals of Eugenics*, 8:376–386, 1938. 1

[8] S. Gao, I. W.-H. Tsang, L.-T. Chia, and P. Zhao. Local features are not lonely - laplacian sparse coding for image classification. In *CVPR*, pages 3555–3561, 2010. 7

[9] X. He and P. Niyogi. Locality preserving projections. In *NIPS*, 2003. 1, 3

[10] C. Kanan and G. W. Cottrell. Robust classification of objects, faces, and flowers using natural image statistics. In *CVPR*, pages 2472–2479, 2010. 6

[11] S. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale l_1-regularized least squares. *IEEE J. Sel. Top. in Sig. Proc.*, 1(4):606–617, 2007. 5

[12] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *NIPS*, pages 801–808. MIT Press, 2006. 5

[13] A. Martinez and A. Kak. PCA versus LDA. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(2):228–233, 2001. 1

[14] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. In *ICCV*, 2001. 1

[15] R. Timofte and L. Van Gool. Sparse representation based projections. In *BMVC*, 2011. 1, 3, 4, 5

[16] T. Tuytelaars, M. Fritz, K. Saenko, and T. Darrell. The NBNN kernel. In *ICCV*, 2011. 7

[17] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 7, 8

[18] J. Wright, A. Y. Yang, A. Ganesh, S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Learning*, 31(2), February 2009. 1, 4, 5

[19] A. Yang, A. Ganesh, Z. Zhou, S. Sastry, and Y. Ma. Fast l_1-minimization algorithms and an application in robust face recognition: a review. Technical Report UCB/EECS-2010-13, University of California, Berkeley, 2010. 5

[20] J. Yang, K. Yu, Y. Gong, and T. S. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, pages 1794–1801. IEEE, 2009. 5, 7

[21] L. Zhang, M. Yang, and X. Feng. Sparse representation or collaborative representation: Which helps face recognition? In *ICCV*, 2011. 1, 5, 7

[22] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *JCGS*, 15:262–286, 2006. 1