Iterative Nearest Neighbors

Radu Timofte, Luc Van Gool

[a] *VISICS, ESAT-PSI / iMinds, KU Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium*
[b] *Computer Vision Lab, D-ITET, ETH Zurich, Sternwartstrasse 7, 8092 Zurich, Switzerland*

Abstract

Representing data as a linear combination of a set of selected known samples is of interest for various machine learning applications such as dimensionality reduction or classification. k-Nearest Neighbors (kNN) and its variants are still among the best-known and most often used techniques. Some popular richer representations are Sparse Representation (SR) based on solving an l_1-regularized least squares formulation, Collaborative Representation (CR) based on l_2-regularized least squares, and Locally Linear Embedding (LLE) based on an l_1-constrained least squares problem. We propose a novel sparse representation, the Iterative Nearest Neighbors (INN). It combines the power of SR and LLE with the computational simplicity of kNN. We empirically validate our representation in terms of sparse support signal recovery and compare with similar Matching Pursuit (MP) and Orthogonal Matching Pursuit (OMP), two other iterative methods. We also test our method in terms of dimensionality reduction and classification, using standard benchmarks for faces (AR), traffic signs (GTSRB), and objects (PASCAL VOC 2007). INN compares favorably to NN, MP, and OMP, and on par with CR and SR, while being orders of magnitude faster than the latter. On the downside, INN does not scale well with higher dimensionalities of the data.

Keywords:
iterative nearest neighbors, least squares, sparse representation, collaborative representation, classification, dimensionality reduction

1. Introduction

It is a common assumption that data has an intrinsic structure which significantly influences the performance of the subsequent applications using it. This is the case especially for classification and dimensionality reduction.

Dimensionality reduction techniques set out to preserve or enhance one or the other property under the reducing projection. Some linear techniques are used for this: Principal Component Analysis (PCA) [1] enhances the concentration of variance, Locality Preserving Projection (LPP) [2] conserves the local affinities, Sparse Representation Linear Projection (SRLP) [3] embeds a sparse representation, and Linear Discriminant Analysis (LDA) [4, 5] maximizes the inter-class vs. intra-class variance. Among the non-linear techniques we have: Locally Linear Embedding (LLE) [6] which preserves the local neighborhood, Laplacian Eigenmaps (LE) [7] and Supervised Laplacian Eigenmaps (SLE) [8] preserving the distances to neighbors, and Sparse Representation Embedding (SRE) [3] preserving the sparse representation.

For classification purposes we have to define those properties and measures used to match a new sample or 'query' against the known, labeled samples or the learned class model. Picking the label of the nearest neighbor as the class label for the query is the simplest such classification decision. 'Nearest' is defined on the basis of some similarity, distance, or metric [9]. The Sparse Representation-based Classifier (SRC) starts from the l_1-regularized least squares decomposition, a Sparse Representation (SR). The decision is based on the class labels of the samples that contribute to the representation of the query. SRC yields state-of-the-art performance in e.g. face recognition [10]. The Collaborative Representation (CR) Classifier (CRC) [11] uses instead the l_2-regularized least squares decomposition.

Apart from the quality of the results, the computational efficiency can be critical in practice. The high performance of SR and CR comes at the price of a computational burden. kNN in comparison is very fast, but less performant.

We aim at combining the high speed of kNN and the performance of SR. Our proposed method is coined the Iterative Nearest Neighbors (INN) representation and is inspired by feedback loop methods. First, INN searches for the nonzero terms in the representation, instead of the weights like most current l_1 solvers would. Each new selection of a sample in the representation is meant to compensate for the errors introduced by the previous selections and, as a consequence, to further reduce the residual between the query and its reconstruction. We have an iterative procedure depicted in Figure 1. Starting from the query sample, INN first gets its nearest neighbor from the training pool of samples. In the next iterations, it each time picks the sample best suited to reduce the residual between the query sample and its reconstruction on the basis of the previously selected samples. The reconstruction is obtained as a linear com-

Email addresses: radu.timofte@vision.ee.ethz.ch (Radu Timofte), vangool@vision.ee.ethz.ch (Luc Van Gool)
URL: http://www.vision.ee.ethz.ch/~timofter/ (Radu Timofte)

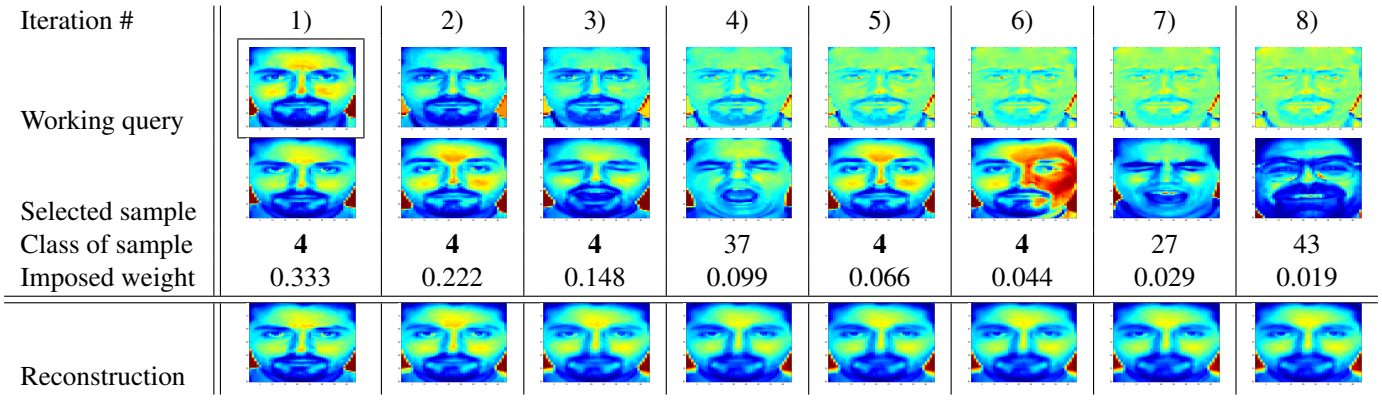


Figure 1: **INN0 algorithm iteration by iteration.** 8 iterations for $\lambda = 0.5$ and $\beta = 0.95$ (95%). At each iteration we show the working query, the selected NN sample from the training pool, its class, the imposed weight, and the current reconstruction. The input query has a summed confidence of 0.813 to belong to class 4.

bination. This process is repeated until a stopping criterion is met. This can be a maximum number of iterations, a residual threshold, or when the sum of the remaining weights in a potentially infinite representation gets too small compared to the sum of the weights already assigned to samples. We transform the selection of each new term into a NN decision by adapting the query at each step by adding the weighted residual introduced by the previous selection. This weight is the regularization parameter of the method. As will become clear, the monotonously increasing sum of the imposed weights in the representation is guaranteed to have a limited value and after a limited number of iterations, the remaining terms can be discarded given an application-specific (error) tolerance to noise.

INN is an iterative procedure and its complexity is dominated by the maximum number of iterations (nonzeroes in the representation) $\times$ the time of the NN 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 newly picked NN. It is comparable to the Lagrangian parameter used in l_1 and l_2-regularized least squares solvers. Nevertheless, tuning the parameter for specific data usually still brings some improvement (as with l_1 and l_2-solvers).

This paper is an extension of our previous work [12] which introduced INN in its original form (in the sequel referred to as INN0). We show INN0 to be a valid approach in cases with moderate amounts of noise, and show that a proposed refinement brings no significant advantages. Moreover, we also extend the empirical validation of the method by comparing it against similar greedy iterative recovery algorithms, *i.e.* Matching Pursuit (MP) [13] and its variant Orthogonal Matching Pursuit (OMP) [14, 15].

The rest of the paper is structured as follows. First, we review and refine the Iterative Nearest Neighbors method in Section 2. We emphasize similarities with other iterative methods by benchmarking its performance and we provide an analysis of its complexity, bounds, and approximations. Then, in Section 3 we derive our INN-based variant of the SRLP dimensionality reduction method, our INN-based Classifier, and INN-based variants for the Naive Bayes classifier and the Sparse Coding Spatial Pyramid Matching method. Section 4 experimentally validates the proposed INN representation and its derived applications, using real-life datasets. The conclusions are drawn in Section 5.

2. Iterative Nearest Neighbors

First we briefly review the main motivation and assumptions we made, then we introduce our INN proposal and provide a theoretical and empirical analysis.

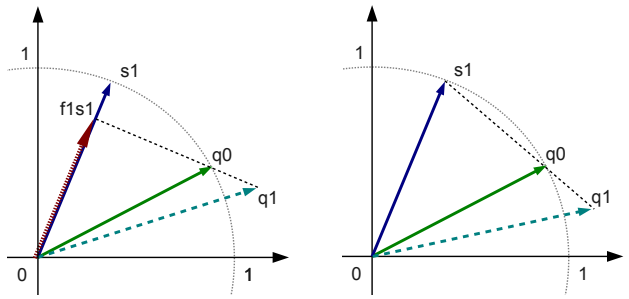
2.1. Motivation and Assumptions

We want to combine the speed and simplicity of kNN or MP with the performance of SR or CR. This is our main motivation. Regarding these properties, we make two observations. Firstly, kNN and MP rely on nearest neighbors and this to a large extent accounts for their speed. Secondly, SR solvers optimize over the weights, and seem to draw much of their performance from the fact that the main supporting samples in the SR representation tend to belong to the data class/cluster of the input query. Thus, we propose a method that combines these elements. Moreover, we introduce a controlling parameter that allows our algorithm to trade off performance for speed.

Three important design choices we make in defining our INN representation are:

1) We try to ensure that the most important samples in the representation are similar to the input query. This motivates the use of nearest neighbor search as main operation and not a regression involving scaling.

2) We envisage a sequential selection process of the samples, with their weights going down as the algorithm proceeds. We will impose guiding weights for the representation and, with these weights steadily decreasing, the importance of each subsequent sample decreases. This said, a specific sample can be selected multiple times, thus even if the weights are predefined discrete values of a decreasing function, the actual weight of such repeatedly selected sample is the sum of the corresponding weights. As a result, the first selected samples are not necessarily the ones with the largest impact (cumulated weight) in the representation.



(a) INN (b) INN0

Figure 2: **INN vs. INN0 first iteration example**: both start with the same initial query sample $\mathbf{q}_0$ and first selected sample $\mathbf{s}_1$, but since INN adapts $\mathbf{s}_1$ using a factor f_1, the updated working queries $\mathbf{q}_1$ differ.

3) We use a working query updated after each selection. The update is an addition of the weighted residual between the working query and the selected sample. In this way: i) the energy (*i.e.* the l_2-norm) of the working query is slowly changing, ii) the working query embeds the history of our selections, iii) the impact of the residual is controlled by a weighting parameter, the same controlling the decay of the weights, and iv) the nearest neighbor to the updated working query will address the cumulated residual caused by the previous selections.

2.2. INN Algorithm

The Iterative Nearest Neighbors (INN) representation is defined by the following recursive procedure. The inputs are a finite sensing matrix (or set) $\mathbf{X} \in \mathbb{R}^{M\times N}$ of bounded training samples $\mathbf{x}_i \in \mathbb{R}^M$ and a new query sample $\mathbf{y} \in \mathbb{R}^M$. We assume $\mathbf{x}_i$ to have unit l_2 norm (however, INN can be easily extended outside this case) and a given maximum number of iterations $K \in \mathbb{N}$ or a reconstruction error $\epsilon \in \mathbb{R}$. The sample $\mathbf{y}$ is approximately decomposed with the help of a series of working queries/vectors $\{\mathbf{q}_k\}$, the first of which $\mathbf{q}_0 = \mathbf{y}$. The INN steps are as follows, starting $k = 1$:

1) Identify the nearest neighbor in $\mathbf{X}$ for the previous $\mathbf{q}_{k-1}$:
$\mathbf{s}_k = NN(\mathbf{X}, \mathbf{q}_{k-1})$
2) Estimate the scaling factor, f_k:
$f_k = \arg\min_{a\in\mathbb{R}} \|\mathbf{q}_{k-1} - a\mathbf{s}_k\|$
3) Update current query $\mathbf{q}_k$ by adding:
$\mathbf{q}_k = \mathbf{q}_{k-1} + \lambda(\mathbf{q}_{k-1} - f_k\mathbf{s}_k)$, with $\lambda > 0$ the residual weight
4) Repeat the steps with $k = k+1$ until $\|\mathbf{y} - \sum_{i=1}^{k-1} \frac{\lambda}{(1+\lambda)^i} f_i\mathbf{s}_i\| \le \epsilon$ or $k > K$.

By construction, we have a recursive process leading to the following formalism, where k is the iteration step, $\mathbf{s}_k$ is the selected support vector, $\mathbf{s}_k = NN(\mathbf{X}, \mathbf{q}_{k-1})$, and f_k an adaptive scaling factor $f_k = f(\mathbf{q}_{k-1}, \mathbf{s}_k)$:

$$\begin{aligned} \mathbf{q}_k &= \mathbf{y}, && k = 0 \\ \mathbf{q}_k &= \mathbf{q}_{k-1} + \lambda(\mathbf{q}_{k-1} - f_k\mathbf{s}_k), && k > 0 \\ &= (1+\lambda)\mathbf{q}_{k-1} - \lambda f_k\mathbf{s}_k, && k > 0 \end{aligned} \tag{1}$$

Now, we can reconstruct the original query by recursive substitution:

$$\begin{aligned} \mathbf{y} &= \mathbf{q}_0 \\ &= (\mathbf{q}_1 + \lambda f_1\mathbf{s}_1)/(1+\lambda) \\ &= ((\mathbf{q}_2 + \lambda f_2\mathbf{s}_2)/(1+\lambda) + \lambda f_1\mathbf{s}_1)/(1+\lambda) \\ &= \tfrac{1}{(1+\lambda)^2}\mathbf{q}_2 + \tfrac{\lambda}{(1+\lambda)^2} f_2\mathbf{s}_2 + \tfrac{\lambda}{(1+\lambda)^1} f_1\mathbf{s}_1 \\ &\dots \\ &= \tfrac{1}{(1+\lambda)^k}\mathbf{q}_k + \textstyle\sum_{i=1}^{k} \tfrac{\lambda}{(1+\lambda)^i} f_i\mathbf{s}_i \end{aligned} \tag{2}$$

Thus, for the reconstruction of the original query $\mathbf{y}$, each selection/support vector, $\mathbf{s}_i$, has a weight $\frac{\lambda}{(1+\lambda)^i} \in (0,1)$, that is imposed by the algorithm. The weight depends on λ and iteration, i. As an approximation of $\mathbf{y}$ one can use the sum of weighted samples $\sum_{i=1}^{k} \frac{\lambda}{(1+\lambda)^i} f_i\mathbf{s}_i$. If one iterates long enough (k sufficiently high), the approximation error $\frac{1}{(1+\lambda)^k}\mathbf{q}_k$ will be small.

Continuously compensating for the residuals at each stage of the algorithm is an idea akin to regulators adding a regulatory value after each feedback loop, leading up to the convergence of the system output. Our regulatory parameter is λ. It determines to what degree the residual vector is used in the adapted query for the next INN iteration.

Considering f constant and equal to 1 results in the INN0 formulation propounded in our original work [12].

Figure 2 (a) and (b) show one iteration for INN and INN0, resp. INN and INN0 diverge in their behavior from the first iteration. As can be seen, INN0 uses the samples as they are (all lying on a unit hypersphere), whereas INN multiplies each sample $\mathbf{s}_k$ with a factor f_k, such that $f_k\mathbf{s}_k$ comes as close as possible to the working query $\mathbf{q}_{k-1}$. The next working query, $\mathbf{q}_k$, is constructed by calculating the residual between $\mathbf{q}_{k-1}$ and the (f_k - weighted) sample, and moving away from $\mathbf{q}_{k-1}$ in the opposite direction over $\lambda(\mathbf{q}_{k-1} - (f_k)\mathbf{s}_k)$. If the residual is small in length - *i.e.* if $(f_k)\mathbf{s}_k$ is really close to $\mathbf{q}_{k-1}$ - then $\mathbf{q}_k$ will remain close to $\mathbf{q}_{k-1}$, thereby increasing the chance that the same close sample $\mathbf{s}_k$ will be chosen again as closest neighbor for a couple of iterations (and with the same f-factors in the case of INN). This will increase that sample's sum of weights. Letting the algorithms the possibility to re-select the same sample a couple of times also allows them to finetune the ratios of the sum of weights of such re-selected samples. On the other hand, if the residual is large, $\mathbf{q}_k$ is chosen far from $\mathbf{q}_{k-1}$. When looking for the nearest neighbor of $\mathbf{q}_k$, one is now exploring quite a different part of the sample space and chances are high that immediately a different nearest sample is found. Hence, if $(f_k)\mathbf{s}_k$ is a bad approximation of the working query, it is quickly left behind for another sample.

As to the difference between INN0 and INN, the extra factors f_k that INN contains, allow it to approach the working queries more closely. As just explained, that would keep the next working query closer to the current one if the same λ is used. Alternatively, λ (and therefore β) could be made somewhat larger for INN than for INN0 while moving over similar distances. That is indeed the choice that we made in the paper.

In Table 1 we list the algorithmic steps of INN0 as well as those of the current, refined version INN for which we consider

Table 1: Greedy iterative signal recovery algorithms, with the typography bringing out the similarity between their steps. The ones that are stricken out have been dropped in our implementations.

	OMP [14, 15]	MP [13]	INN0 [12] (ours)	INN (ours)
Input:	measurements $\mathbf{y} \in \mathbb{R}^M$, sensing matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$, maximum iterations K, allowed error $\epsilon \in \mathbb{R}_+$,		and l_2 normalized columns, and β,	
Initialize:	iteration counter $k = 0$, residual vector $\mathbf{r}_0 = \mathbf{y}$,		and $\lambda = e^{-\frac{\log_e(1-\beta)}{K}} - 1$, and working vector $\mathbf{q}_0 = \mathbf{y}$,	
While	$\|\mathbf{r}_k\|_2 > \epsilon$ AND $k < K$ **do**	$\|\mathbf{r}_k\|_2 > \epsilon$ AND $k < K$ **do**	~~$\|\mathbf{r}_k\|_2 > \epsilon$ AND~~ $k < K$ **do**	~~$\|\mathbf{r}_k\|_2 > \epsilon$ AND~~ $k < K$ **do**
	$k = k + 1$	$k = k + 1$	$k = k + 1$	$k = k + 1$
(*Identification*)	$\mathbf{s}_k = \arg\max_{\mathbf{x}_i} \|\mathbf{x}_i^T \mathbf{r}_{k-1}\|$	$\mathbf{s}_k = \arg\max_{\mathbf{x}_i} \|\mathbf{x}_i^T \mathbf{r}_{k-1}\|$	$\mathbf{s}_k = \arg\max_{\mathbf{x}_i} \mathbf{x}_i^T \mathbf{q}_{k-1}$	$\mathbf{s}_k = \arg\max_{\mathbf{x}_i} \|\mathbf{x}_i^T \mathbf{q}_{k-1}\|$
(*Estimation*)	$\mathbf{a} = \arg\min_{\mathbf{u}} \|\mathbf{y} - [\mathbf{s}_1, \cdots, \mathbf{s}_k]\mathbf{u}\|_2$	$a_k = \mathbf{s}_k^T \mathbf{r}_{k-1}$		$f_k = \mathbf{s}_k^T \mathbf{q}_{k-1}$
(*Update*)			$\mathbf{q}_k = \mathbf{q}_{k-1} + \lambda(\mathbf{q}_{k-1} - \mathbf{s}_k)$	$\mathbf{q}_k = \mathbf{q}_{k-1} + \lambda(\mathbf{q}_{k-1} - f_k \mathbf{s}_k)$
	$\mathbf{r}_k = \mathbf{y} - [\mathbf{s}_1, \cdots, \mathbf{s}_k]\mathbf{a}$	$\mathbf{r}_k = \mathbf{r}_{k-1} - a_k \mathbf{s}_k$	~~$\mathbf{r}_k = \mathbf{r}_{k-1} - \frac{\lambda}{(1+\lambda)^k} \mathbf{s}_k$~~	~~$\mathbf{r}_k = \mathbf{r}_{k-1} - \frac{\lambda}{(1+\lambda)^k} f_k \mathbf{s}_k$~~
End				
Output:	$\{(a_k, \mathbf{s}_k)\}_{k \in \{1,\cdots,K\}}$	$\{(a_k, \mathbf{s}_k)\}_{k \in \{1,\cdots,K\}}$	$\{(\frac{\lambda}{(1+\lambda)^k}, \mathbf{s}_k)\}_{k \in \{1,\cdots,K\}}$	$\{(\frac{\lambda}{(1+\lambda)^k} f_k, \mathbf{s}_k)\}_{k \in \{1,\cdots,K\}}$

f_k to be the scalar product between the selection vector and the working query vector, $f_k = \mathbf{s}_k^T \mathbf{q}_{k-1}$. Moreover, for the NN selection we take the scalar product. We considered also the l_1-norm and l_2-norm. The first assigns equal importance to all feature vector elements, the second penalizes the big differences more, while the scalar/inner product correlates well with l_2 ranking when the vectors have unit l_2 norm. Both choices for f_k and NN are similar to those in Matching Pursuit (MP) [13, 16], where the residual is actively reduced after each iteration. Orthogonal Matching Pursuit (OMP) [14, 15] builds upon the idea of MP and refines the estimated reconstruction by using all the selections from the previous iterations. MP and OMP belong to the same type of greedy iterative signal recovery algorithms to which our INN0 and INN also belong.

The stopping criterion of MP and OMP can be the maximum number of iterations K or the remaining residual error ϵ. For INN and INN0 we decided to use only the maximum number of iterations K as stopping criterion in all our experiments. Figure 1 shows how the INN0 algorithm constructs the sparse representation for a face sample, iteration by iteration. The difference between INN0 and INN is that for INN we perform a refinement of the weight after the nearest neighbor selection was made to further reduce the residual, while for INN0 we do not (see Figure 2). This refinement gives more flexibility and a performance gain to INN with respect to INN0, especially in (close to) noiseless conditions (as shown in the next Section).

According to Table 1, INN resembles MP – both being greedy iterative algorithms, making NN selections and computing scalar products. MP actively reduces the residual energy (or l_2 norm, or error) by working directly on the residuals. The residual is exponentially decaying and MP is prune after a few iterations to make selections correlated with a very low energy residual, less reliable and easily sunk in or influenced by the (measurement) noise. OMP inherits this problem of MP. For INN (and INN0) we do not have this problem, since INN prescribes the weights and employs working queries that are (slowly) diverging (in energy) from the input sample, with the final result of reducing the residual energy. Moreover, the behavior of INN is controlled by a weighting parameter. These differences between INN and MP are responsible for INN's better performance under similar conditions, as shown next.

2.3. Theoretical Analysis/Bounds

The recursive procedure of the INN algorithm brings us to the following optimization problem for which it provides an approximate solution:

$$\arg\min_{\{\mathbf{s},f\}_{i=1}^{\infty}} \|\mathbf{y} - \sum_{i=1}^{\infty} \frac{\lambda}{(1+\lambda)^i} f_i \mathbf{s}_i\| \tag{3}$$

where the selections $\mathbf{s}_i$ and scaling factors f_i (depending on the query $\mathbf{q}_{i-1}$ and selection $\mathbf{s}_i$) are determined as to minimize the residual $\|\mathbf{q}_{i-1} - f_i \mathbf{s}_i\|$, and where the $\mathbf{q}_{i-1}$ are defined by eq. (1).

The maximum sum of the imposed weights is 1:

$$\lim_{K \to \infty} \sum_{i=1}^{K} \frac{\lambda}{(1+\lambda)^i} = 1 \tag{4}$$

and this regardless of the exact value of $\lambda > 0$ and the selection of $\mathbf{s}_i \in \mathbf{X}$. With a bound on $|f_i| < \bar{f}$ for the INN selections, we have:

$$\sum_{i=1}^{\infty} |\frac{\lambda}{(1+\lambda)^i} f_i \mathbf{s}_i| \leq \bar{f} \max_{\mathbf{x} \in \mathbf{X}} |\mathbf{x}| \sum_{i=1}^{\infty} \frac{\lambda}{(1+\lambda)^i} = \bar{f} \max_{\mathbf{x} \in \mathbf{X}} |\mathbf{x}| < \infty \tag{5}$$

and, therefore, the INN representation is absolute convergent and convergent to a finite vector, not necessarily $\mathbf{y}$.

The number K of terms necessary to sum up to a β-fraction of the total sum of the imposed weights is given by:

$$\beta = (\sum_{i=1}^{K} \frac{\lambda}{(1+\lambda)^i})/1 = 1 - (1+\lambda)^{-K} \tag{6}$$

and hence

$$K = \lceil -\frac{\log(1-\beta)}{\log(1+\lambda)} \rceil \tag{7}$$

and

$$\lambda = e^{-\frac{\log(1-\beta)}{K}} - 1. \tag{8}$$

The most important weights are the first K. We can safely search only for the first K terms in the query expansion and obtain a large drop in computational cost. In practice, $\beta < 1$ is fixed and we compute either K based on λ or λ based on the number K of iterations/maximum support vectors one can afford to have.

2.4. Empirical Performance

While many vision applications nowadays handle (very) large datasets, we are not always in control of key aspects such as the sparsity of the signals or the level of noise. Thus, we chose to first observe the empirical performance of our method through computer simulations. The most common scenarios in vision have l_2-normalized (histogram) data. This data is often nonnegative. Another assumption is (for example, in face recognition) that new samples can be sparsely decomposed over the prior known data. Our methods (INN0 and INN) are meant for such cases. Moreover, in the case of INN0 the weights have to be nonnegative and ideally sum up to 1. INN is more flexible, accommodating cases with negative weights, since the f_k scaling weights in this representation are allowed to be negative (while the imposed guiding weights are still positive and sum up to 1).

In our experiment we measure the effectiveness of the sample selection also known as support vector recovery by checking the fraction of support vectors in the sparse representation compared to the number of terms in the linear combination creating the query sample. This is indicative for how likely the method provides the desired (structural) information for further tasks such as classification or affinity computation in dimensionality reduction. Ideally, we want a perfect match between the selected/support vectors provided by the method and the ones known to the computer to (sparsely) compose the query with zero reconstruction error. Our INN0 and INN methods are approximation methods not guaranteed to converge to a perfect reconstruction, and we do not use the reconstruction error but the coefficients in the representation for further applications. If all the correct samples are picked by the INN solution then a further least squares fit would be able to retrieve the perfect reconstruction.

In Table 1 are depicted, side by side, the algorithmic steps of the methods considered in this empirical evaluation:

1) Orthogonal Matching Pursuit (OMP) [14, 15].

2) Matching Pursuit (MP) [13].

3) Iterative Nearest Neighbors algorithm 0 (INN0), as introduced in [12], $\beta = 0.95$.

4) Iterative Nearest Neighbors algorithm (INN), enhanced version of INN0, as proposed here, $\beta = 0.99$.

We compare greedy iterative algorithms with a strict control over the maximum number of support vectors (samples) in the solution. By fixing the number of iterations we in fact fix the maximum number of support vectors/ nonzeros for the reconstructed solution.

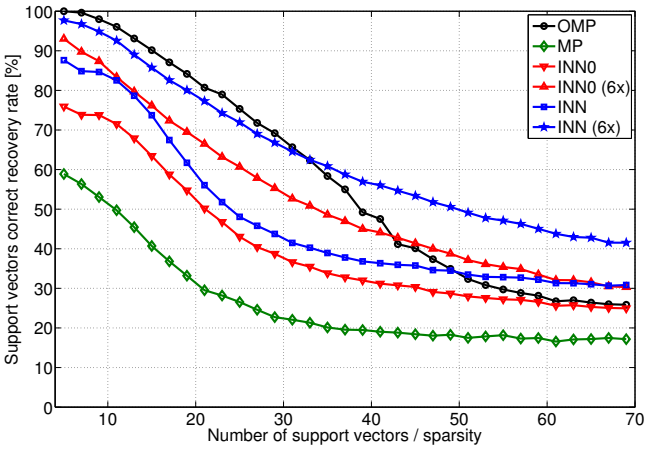


Figure 3: Support vectors correct recovery rate [%] for a K-sparse Gaussian signal vector as a function of sparsity K. All the methods were run for K iterations, with the exception of INN (6x) and INN0 (6x), which were allowed to run 6 times longer, 6xK. Noiseless conditions.

For each trial, we construct an $M \times N$ ($M = 128$ and $N = 512$) $\mathbf{X}$ sensing matrix with entries drawn independently from a Gaussian distribution $\mathcal{N}(0, \frac{1}{M})$. Moreover, the N samples are taken in absolute value and l_2 normalized. Additionally, we generate a K-sparse vector $\mathbf{y}$ whose support vectors are randomly chosen from the columns of $\mathbf{X}$. The nonzero coefficients are drawn from a standard Gaussian, then again absolute values are taken and the result is l_1 normalized. We consider 4 levels of Gaussian noise: none, low $\mathcal{N}(0, \frac{0.3}{M})$, high $\mathcal{N}(0, \frac{1}{M})$, and extreme $\mathcal{N}(0, \frac{5}{M})$. The noise is sampled individually for each query and added to it. For each recovery algorithm we perform 500 independent trials for each K-sparsity and plot the average correct recovery rate of the support vectors from the input queries, as well as the running time and average number of nonzeros from the reconstructions. Note that the algorithms are run for a maximum of K iterations, thus imposing maximum K nonzeros in the solution. This is an oracle decision, however. We want to have a large percentage of the correct support vectors already recovered in the solution once K iterations are reached. If more iterations are used it is likely that more undesired support vectors are selected for the solution.

In Figure 3, we provide the correct support vector recovery rate performance as a function of the sparsity level K. The corresponding running times per each query and the average number of nonzeros used in the reconstructed solution are provided in Figure 4 and Figure 5, resp. The simulation results show that OMP has the highest accuracy for sparsities up to 40, while MP has the lowest. OMP greatly benefits from its orthogonal selection, at each iteration selecting a new support vector. Our INN0 and INN algorithms have an intermediate behavior for sparsity up to 40, and get on par or better than OMP above 40. If we consider the running time, see Figure 4, we see that OMP is much slower than the other methods, as it employs a least squares estimate at each iteration.

While for OMP the number of iterations determines the sparsity of the solution, the other methods (INN, INN0, MP) can repeatedly pick up the same sample at different iterations. Also,

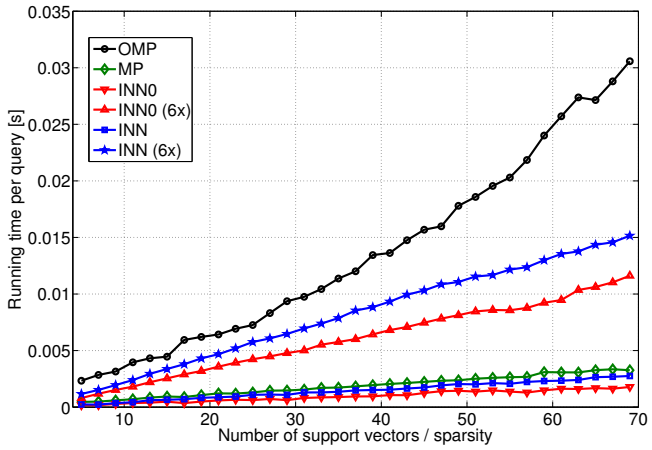


Figure 4: Running time (in seconds) per query as a function of sparsity K. All the methods are run for K iterations, with the exception of INN (6x) and INN0 (6x), which are allowed to run 6 times longer, 6xK. Noiseless conditions.

the number of iterations directly impacts on the running time, and generally increases the number of selected samples in the solution and the recovery rates.

If we allow for 6x more iterations for INN and INN0, we are still up to 2 times faster than OMP, but the performance improves. INN (6x) has a performance on par with OMP up to a sparsity of 30, and better after. At the same time, from Figure 5, we see that sparsity estimation by INN0 (6x) and INN (6x) is reasonable with less than double what it should be.

For MP (6x) the situation is very different, as it fails to achieve the intended sparsity by far (more than 4 times what it should be). Therefore, its results were not included. Overall, INN improves over INN0. Both algorithms are at the MP speed level and closer to the OMP performance level, especially when more iterations are allowed or the number of ground truth support vectors is high. OMP is 6 times up to more than an order of magnitude slower than INN, INN0 and MP.

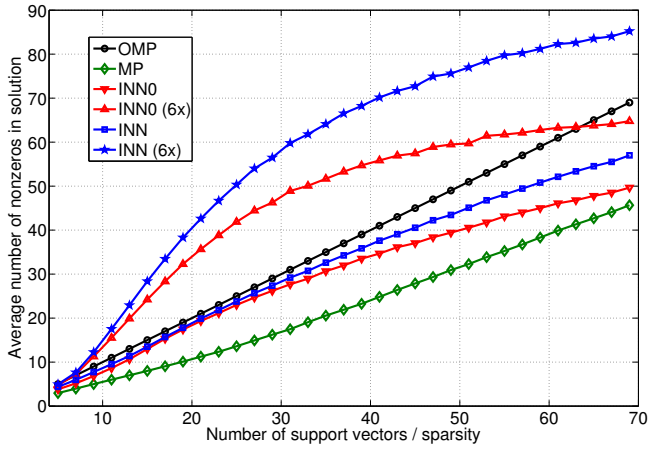


Figure 5: Average number of nonzeros used in the reconstruction as a function of sparsity K. All the methods are run for K iterations, with the exception of INN (6x) and INN0 (6x), which are allowed to run 6 times longer, 6xK. MP with 6 times more iterations picks more than double the desired number of support vectors and is not included. Noiseless conditions.

In real computer vision applications there is noise. In Figure 6 we depict the influence of noise on the performance. From these simulations we conclude that OMP copes with noise much worse than the other methods. In particular INN and INN0 can outperform OMP by a margin, and at the same time be much faster as we will show in the experimental section for classification.

2.5. Structured Representations

INN provides an approximation of the problem in eq. (3). The geometrical progression of the imposed weights enforces the sparsity of the representation, after a number (imposed or derived using eq. (7)) of K selections the imposed weights being too small and thus the negligible impact of the further selections in the representation.

INN builds a special structured representation in that it imposes (guiding) weights/coefficients and optimizes over the (fixed) pool of samples/atoms. This is unlike other schemes like the sparse solvers, where such a restriction on the choice of the weights is not applied. While the idea of imposed weights may be counter-intuitive, it brings important advantages, as we show in our experiments.

Put in different words, the INN algorithm selects a sample at each iteration and updates the weight with a decreasing value defined by λ and the iteration number i, $\frac{\lambda}{(1+\lambda)^i} \in (0, 1)$. The same samples can be selected multiple times during the procedure, and the guiding weights assigned to them are combinations of a discrete, known set with cardinality equal to the maximum number of iterations. The higher the number of iterations, the smaller the weights.

The reconstruction is guaranteed to converge (not necessarily to the query sample) in the limit. One can abort the INN procedure as soon as the representation residual or the sum of the remaining weights is less than the data noise. The INN algorithm corresponds to a constrained recurrent residual decomposition with imposed guiding weights and, thus, sparsity.

By imposing the maximum number of iterations we impose the maximum number of nonzero samples / support vectors in the INN representation. If we also consider an approximation tolerance as a percentage of the completely expanded sum of imposed weights, then using the relation (7) yields the regulatory parameter λ.

In the following sections, most of the derivations while made for INN are also valid for INN0. In our further experiments, for INN0, the percentile is set to 95% ($\beta = 0.95$). The representation tolerance, corresponds to the remaining 5%. For the refined INN, the percentile is set to 99% ($\beta = 0.99$).

3. Applications

Like the other representations, INN can be integrated in many different computer vision applications. In this section, we apply INN for dimensionality reduction and classification tasks. In both applications, INN replaces SR and provides an alternative to other representations at various levels. The INN0 derivations are obtained similarly to INN.

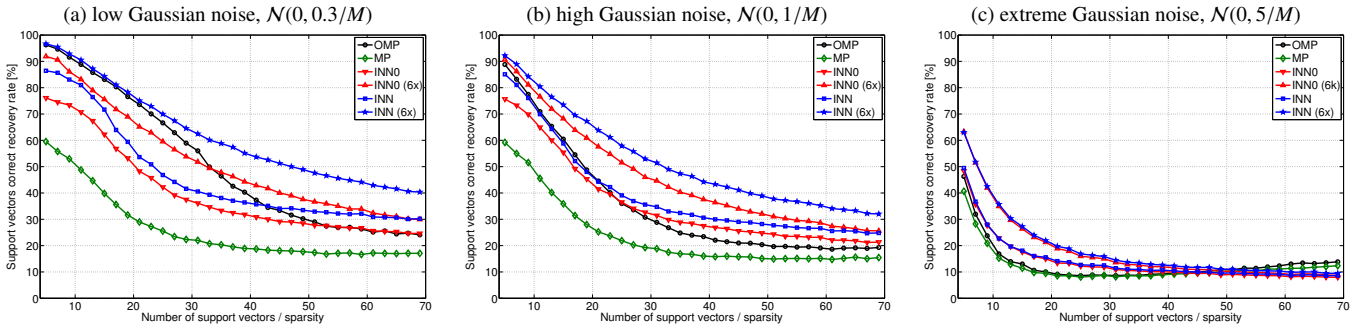


Figure 6: The effect of Gaussian noise on support vectors correct recovery rate [%] for K-sparse Gaussian signal vector as a function of sparsity K. The noise was added only to the query K-sparse signal.

3.1. Dimensionality Reduction

Recently, a Sparse Representation - based Linear Projection (SRLP) method has been proposed [3]. It is a variant of Locality Preserving Projections (LPP) [2] where the affinities/weights are substituted by the absolute coefficients of the l_1-regularized least squares solution. For classification SRLP was shown to improve over the original LPP and Linear Discriminant Analysis (LDA) projections [3].

We derive our INN-based Linear Projection (INNLP) by using the framework of SRLP where we change only the weights. Now the weights are obtained by solving the INN representation. For introducing our INNLP variant, we closely follow the SRLP exposition from [3].

Both SRLP and our INNLP variant aim at embedding the weights from the sparse representation as affinity scores in the linear embedding space. The main steps are: adjacency graph construction, weight computation, and eigenmapping.

INNLP (and SRLP) replaces the graph representation with a symmetric adjacency matrix of weights $\mathbf{W}_{N\times N}$ determined by the corresponding sparse representations. For each training sample $\mathbf{x}_i$ we compute the INN representation in terms of the other samples. Let $\mathbf{w}_i \in \mathbb{R}^N$ be these INN weights, where $\mathbf{w}_{ij}$ is the contribution of $\mathbf{x}_j$ to the INN representation of $\mathbf{x}_i$. In a supervised scenario we know the labels and compute the INN representation restricted to the samples sharing the same class label with the query. The INNLP weighted symmetric adjacency matrix is then taken as

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

where we use the magnitude of the INN coefficients/weights.

As with the original LPP and SRLP, the eigenmapping step 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, $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ji}$, $\mathbf{L}$ is the Laplacian matrix, $\mathbf{L} = \mathbf{D} - \mathbf{W}$, and $\mathbf{X}$ is the matrix with the training samples, $\mathbf{x}_i$, as columns. Let $\mathbf{a}_1, \cdots, \mathbf{a}_P$ be the solutions of (10) sorted in ascending order of their corresponding eigenvalues. The embedding is defined by

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

where P is the dimension of the embedding space.

In a similar fashion one can replace the sparse representation from the non-linear Sparse Representation-based Embedding [3] or other graph-based dimensionality reduction methods to derive INN-based variants.

3.2. Classification with INN

3.2.1. INN-based Naive Bayes Image Classifier

The Naive Bayes Nearest Neighbor (NBNN) image classifier has been advocated in [17]. It proved a powerful classifier and in its original form, to a certain extent, it is a learning-free and parameter-free method. NBNN avoids discretization and generalizes well by using 'image-to-class' instead of 'image-to-image' comparisons.

Here we use two Naive Bayes variants, one based on INN representations and one on SR representations: the NBINN - INN-based Naive Bayes classifier and NBSR - SR-based Naive Bayes classifier. In [18] a study of such Naive Bayes variants was already presented.

The Naive Bayes classifier [17] comes with strong independence assumptions. A query image is defined 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 decision is

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

where $\mathbf{C}$ is the set of class labels.

For a given feature sample $\mathbf{q}$ and a matrix $\mathbf{X}$ with N labeled samples stacked column-wise, $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$, NBSR optimizes over $\mathbf{w}$ to obtain the SR representation:

$$\min_{\mathbf{w}} \|\mathbf{q} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_1 \tag{13}$$

The absolute values of the SR coefficients reflect their importance for the query $\mathbf{q}$ and are further l_1-normalized to sum up to 1:

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

NBINN follows the same steps as NBSR, but substitutes SR by INN. The likeliness of $\mathbf{q}$ to belong to class c is

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

where $\mathbf{X}_c$ is the $\mathbf{X}$ matrix constrained to the columns (feature samples) corresponding to class c.

The classification decision for both NBSR and NBINN is taken using:

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

The query image represented by $\mathbf{Q}$ is assigned to the class with the largest cumulated importance/likeliness in the representations of $\mathbf{q} \in \mathbf{Q}$.

3.2.2. INN-based Classifier

The Sparse Representation-based Classifier (SRC) [10] has been proposed for face recognition. That formulation explicitly deals with noise and uses the residuals for taking the classification decision. In our experiments we do not have the explicit noise handling. To also construct a classifier out of OMP we substitute SR with the OMP representation and use the residuals for decision taking. We call this the OMP-based Classifier (OMPC).

For the INN-based Classifier (INNC) we replace SR with our proposed INN representation and the decision is taken based on cumulating the nonzero coefficients in absolute value as importance indicators for each query sample, as in [3].

Given a new query sample $\mathbf{q} \in \mathbb{R}^M$ we compute the INN representation over the training samples. Let $\mathbf{v} \in \mathbb{R}^N$ be the coefficients of the INN sparse representation in absolute value and further normalized to sum up to 1. The INNC decision is taken as:

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

where $t_i \in \mathbf{C}$ is the class label and $\mathbf{v}_i$ corresponds to the weight of the i-th sample, $\mathbf{x}_i$, in the INN representation. For the query sample $\mathbf{q}$, $\hat{c}$ is the class with the largest impact in the INN representation.

3.2.3. INN Spatial Pyramid Matching

The Sparse Coding Spatial Pyramid Matching (ScSPM) method has been proposed for image classification in [19]. Sparse coding uses a l_1-regularized least squares formulation and is seen as a 'soft assignment' over the learned dictionary, in contrast to a nearest neighbor 'hard' approach.

We test the versatility and power of our INN representation by directly replacing the original sparse coding in ScSPM with our INN representation only in the testing part. For the details and the implementation of the ScSPM framework, the reader is referred to the original work [19]. In our experiments, the INN Spatial Pyramid Matching variant is called INNSPM.

4. Experimental Results

This section describes and discusses experimental results. First, we evaluate the influence of the parameters on the classification performance. Then, we compare different dimensionality reduction methods in conjunction with a battery of classifiers. For classification we use benchmarks ranging from face and traffic sign recognition to scene classification and image retrieval. Finally, we summarize the strengths and weaknesses of our proposed INN representation and its applications.

4.1. Parameters

The regularization parameter λ controls the performance of our INN representation (the number of iterations K being derived using eq. (7)). To assess its influence we use the INNC classifier and compare with a set of other representation-based classifiers: NN, INN0C, SRC, CRC, OMPC. The power of the representations transpires from the the performance of the classifiers. Each representation is controlled by a (regulatory) parameter comparable to our λ in INN. The classification performance and the running time depends on this parameter. The results are presented in Figure 7. Next, we introduce the benchmark and the methods used, then we discuss the results and consider the impact of approximations.

4.1.1. Benchmark

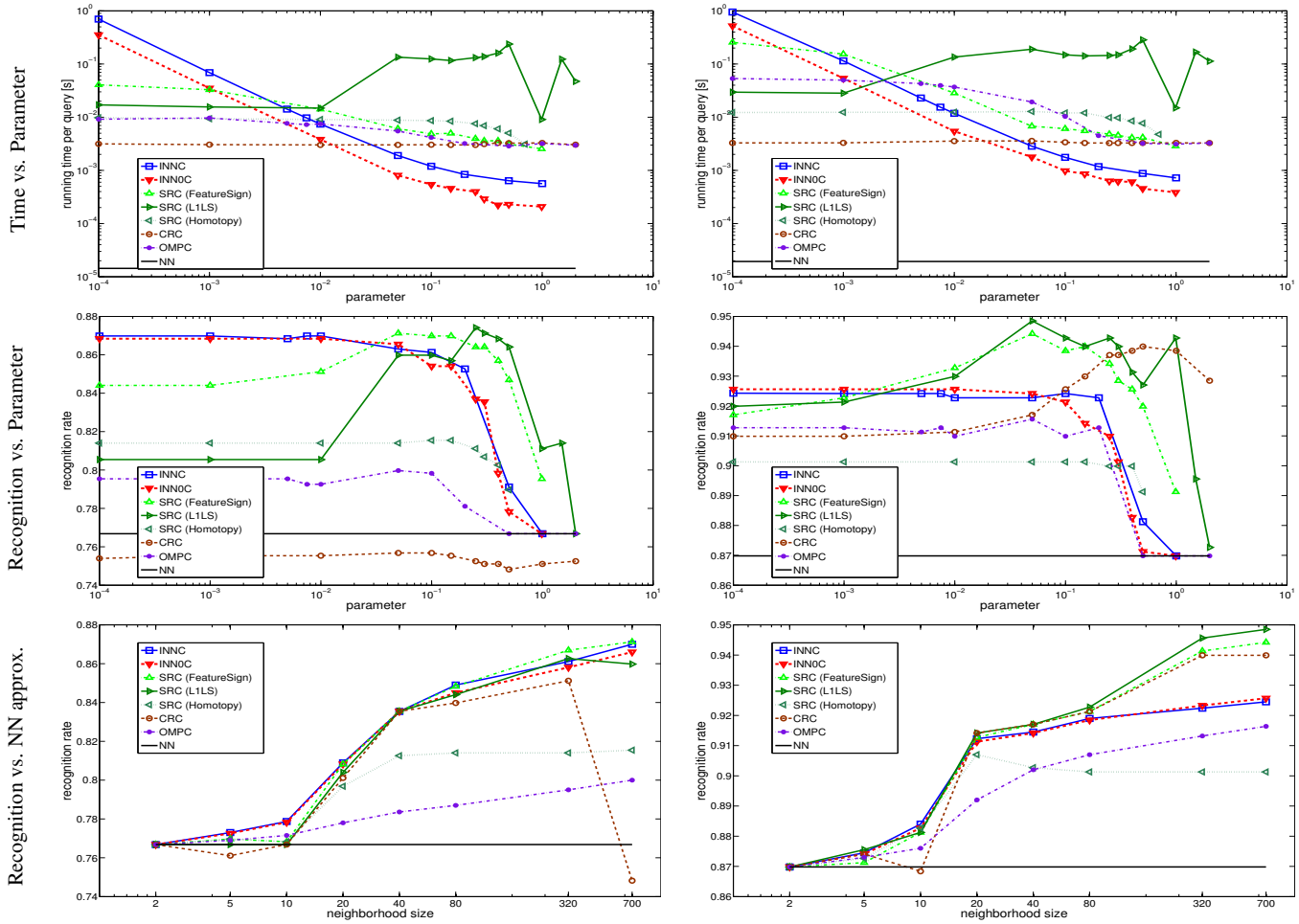
We use the AR benchmark as in [11], *i.e.* only the subset of expression and illumination changes of the original face dataset [20]. There are 100 individuals and for each we have 7 images for training from Session 1, and 7 for testing from Session 2. The images are cropped to 60x43 pixels. A raw image is represented by the vector containing the gray levels of the pixels in lexicographic order. We use the Regularized LDA [4, 5] projected features with the RLDA parameter fixed to 0.001. We chose two operating points (at 30-dimensional embeddings and 99-dimensional embeddings) to capture the behavior of the classifiers in low and high dimensional spaces (with respect to the features – having 100 classes, 99 is the maximum number of dimensions for an RLDA embedding). In our experiments, all features are l_2-normalized.

4.1.2. Compared Classifiers

Under the same conditions, we report in Figure 7 the impact of the parameters on the proposed INNC and INN0C classifiers (see Section 3.2.2), the Sparse Representation Classifier (SRC) [10], the Collaborative Representation Classifier (CRC) [11], and OMP (see Section 3.2.2). As a reference, we provide the results obtained with the simple nearest neighbor classifier (NN).

The performance controlling parameters varied in Figure 7 are the λ residual weight parameter for INNC and INN0C (see Section 2.2), the regularization λ parameter for SRC from eq. (18), the regularization λ parameter for CRC from eq. (20), and the allowed reconstruction error ϵ for OMPC (see Table 1).

In practice, a measurement noise level or a representation tolerance is assumed a priori, as we do for our INNC classifier introduced in Section 3.2.2. The INN0C is derived in the same way as INNC. For our INN representation the percentage of the sum of weights for the full decomposition is fixed to 99% ($\beta = 0.99$) and the number of INN iterations is determined by eq. (7) for a specific value of the parameter λ. For INN0 we use $\beta = 0.95$ and eq. (7).



a) 30-dimensional LDA projection

b) 99-dimensional LDA projection

Figure 7: Parameter influence on AR dataset with low, 30-dimensional LDA projections and relatively high 99-dimensional LDA projections. Independent of the dimensionality of the embedding, usually INN0C and INNC are well-behaved w.r.t. the parameter λ, i.e: the lower λ is, the better the performance. The reported time for CRC does not include the pre-computation of the projection matrix. For OMPC, the parameter is the allowed OMP reconstruction error, ϵ in Table 1, while for the SRC and CRC is their corresponding regulatory parameter λ.

For SRC [10], we need to solve the following SR problem:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \|\mathbf{q} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_1 \quad (18)$$

where λ is the regularization parameter and the query sample $\mathbf{q}$ is assigned to:

$$\hat{c} = \arg\min_{c\in\mathbf{C}} r_c, r_c = \|\mathbf{q} - \mathbf{X}_c\hat{\mathbf{w}}_c\|_2, \quad (19)$$

where r_c is the residual corresponding to class c samples, $\mathbf{X}_c$, and their respective coefficients $\hat{\mathbf{w}}_c$. We used three l_1-regularized least squares solvers for the SR problem: L1LS [21], Homotopy (Hmtp) [22, 23], and Feature Sign (FeSg) [24]. The reader is referred to [25] for a comparison of these l_1-solvers. We use a reconstruction error tolerance of 0.05, for the FeatureSign, L1LS, and Homotopy solvers, similar to [10, 3, 25].

For CRC [11], we solve the ridge regression formulation

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \|\mathbf{q} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2 \quad (20)$$

with algebraic solution

$$\hat{\mathbf{w}} = \mathbf{P}\mathbf{q}, \mathbf{P} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T, \quad (21)$$

and assign $\mathbf{q}$ to class

$$\hat{c} = \arg\min_{c\in\mathbf{C}} r_c, r_c = \|\mathbf{q} - \mathbf{X}_c\hat{\mathbf{w}}_c\|_2/\|\hat{\mathbf{w}}_c\|_2, \quad (22)$$

where r_c is the regularized residual corresponding to class c samples, $\mathbf{X}_c$, and their respective coefficients $\hat{\mathbf{w}}_c$. By adding a small value λ to the diagonal of $\mathbf{X}^\mathbf{T}\mathbf{X}$, the solution is stabilized. This value is the regularization parameter for CRC. We refer to the original work [11] for more details, and to [26, 27] for adaptive and weighted CRC formulations.

4.1.3. Recognition vs. parameter

From Figure 7 we see that the SRC variants with L1LS and FeatureSign solvers come out to be the top performers when their regularization parameter $\lambda \in [0.01, 1]$. INNC and INN0C are similar, providing on par performance with SRC on low dimensional projections, and generally on par with or better than CRC, OMPC, and NN for large λ intervals.

For INNC, the lower λ is, the better the recognition rate but also the slower the computation. INN and INN0 exhibit a well-behaved performance with respect to the λ parameter. Note that the performance gain is negligible if we let λ be less than 0.05 (which corresponds to 62 iterations using (7)). For λ close to 1 and above, the performance of INNC is reduced to that achieved by the Nearest Neighbor (NN) classifier. For $\lambda = 0.25$ the INN0C performance is still very close to its best recognition rate but requires 14 iterations, and thus, only 14 times the time for a NN search.

Setting the appropriate value for the parameter λ is critical for the best performance of all the classifiers studied here, except NN. We observe 'safety intervals' for λ, *i.e.* INNC gets close to its best performance for $\lambda \in (0, 0.1]$, and SRC with FeatureSign for $(0.01, 0.4]$.

4.1.4. Time vs. parameter

As shown in Figure 7, SRC (FeatureSign) benefits greatly from enforced sparsity (large λ), and is able to come close to CRC in terms of running time. For CRC the 'offline' time for computing the projection matrix $\mathbf{P}$ from (21) can be substantial for large training sets. One can use the pseudo-inverse to alleviate this problem [26].

Our proposed INNC and INN0C classifiers are able to achieve on par performance with the best SRC (L1LS) and SRC (FeatureSign) classifiers, but significantly improve the speed. In particular, they are one order of magnitude faster than FeatureSign and up to 3 orders than L1LS. The CRC 'online' time is independent of λ. However, our INN0-based classifier can be more than 10 times faster than CRC, largely due to the fact that CRC uses the residuals and INN0C the weights for taking the decision.

NN, as expected, is by far the fastest method. INNC with K iterations is roughly K times slower than NN, since each INNC iteration involves one NN search.

INNC is slightly slower than INN0C for comparable results. For the same λ parameter, INN requires a larger number of iterations than INN0, due to its larger β and eq. (7), but at the same time INNC performs on par with or slightly better than INN0C.

4.1.5. Approximations

Another important experiment is the one showing how the approximations affect the performance of the classifiers. For speed, the representations are approximated by computing them in a local neighborhood around the query and not over the whole pool of training samples. This case is depicted in the last row of Figure 7, where for each classifier we use their best λ parameter. As expected, the approximation impacts on the performance. The smaller the neighborhood size of training samples, usually the less tight the approximation and the lower the performance of the classifiers. INNC's performance is on par with or better than that of the best SRC classifier, for small sizes (≤ 80) of the neighborhood. Also, is noteworthy that by using a neighborhood of size 80, most classifiers are already in 2% from their top recognition performance using all the pool samples. Figure 7 shows also that effective use is made of the additional samples when available by all methods.

4.2. Dimensionality Reduction

4.2.1. Benchmark

For our dimensionality reduction experiments we use the AR faces benchmark [11, 20] and the GTSRB traffic sign benchmark [28]. For AR we keep the same settings as before (see Section 4.1). GTSRB has a total of 43 classes with 39209 images for training and 12630 images for testing. The annotations were cropped and normalized to 28×28 grayscale pixel patches. The linear projections are learned on the training set, using these grayscale l_2-normalized features in a supervised setting. The projected features are further l_2-normalized before applying the different classifiers: NN, INNC, SRC, CRC – as introduced previously –, Support Vector Machines (SVM) with Linear kernel (L), Intersection kernel (IK), Polynomial kernel

(POLY) and Radial Basis Function kernel (RBF). The polynomial kernel used here is $(\mathbf{x} \cdot \mathbf{y} + 1)^5$ and the RBF one is $\exp(-\|\mathbf{x} - \mathbf{y}\|^2)$.

4.2.2. Methods

We compare our INNLP method for dimensionality reduction versus SRLP [3], Eigenfaces [11], and LDA [4, 5]. Note that the Eigenfaces were used in [11] in the context of face images. We preserve their formulation and name even if we apply them to other classes such as traffic signs. We use the regularized variants of INNLP, SRLP, and LDA. The regularization parameter is set to 0.001 to avoid singularities in all the projection methods. For computing the INN representations in INNLP we set $\lambda = 0.05$ and for the SR representations in SRLP we use the Feature Sign solver with $\lambda = 0.05$.

Note that LDA, the supervised SRLP and INNLP are what we name supervised discriminant embedding (or projection) methods. They are supervised because they use the class information of the training samples. LDA enhances the separability between samples from different classes while SRLP and INNLP embed sparse representations obtained within the same class. In contrast, PCA and Eigenfaces are unsupervised methods, they do not use the class information of the samples.

4.2.3. Recognition vs. dimensionality

Table 2 shows the face recognition results when the embeddings are 5, 10, 30, 54, 99, 120, and 300-dimensional respectively. The choice of the embedding dimensionality is such to cover low, medium and high (relative) dimensionalities for the different projection methods and to assess their effect on the classifiers.

LDA, SRLP, and INNLP are comparable or better than Eigenfaces at an equal embedding dimensionality, as expected, since Eigenfaces are unsupervised projection features. While providing performance on par with the SRLP_{FeSg} projections for all the classifiers, the INNLP projection learning is more than 10 times faster.

Noteworthy is that both INNC and INN0C work best for low-dimensional embeddings (5, 10, 30), regardless the type of projection, but are outperformed by SRC or CRC for higher dimensionalities, more clearly in the case of Eigenfaces.

Table 3 shows traffic sign recognition results for 99-dimensional INNLP, 300 Eigenfaces, 99 SRLP, and 42 LDA projected features. Once more, INNLP is on par with SRLP_{FeSg}, but learns faster. INNLP learns more than ten times faster than SRLP_{Hmtp} using the Homotopy solver. Note that NN performs better with the LDA projections than with the INN or SR-based projections. Also, the SVM classifiers give good performance when using eigenfaces but usually perform worse than the classifiers based on sparse representations (INNC, INN0C, SRC, OMPC) when using discriminant projections.

4.2.4. Performance scaling

As already seen from the results in Figure 7 and Table 2, INN does not scale very well with a higher dimensionality of the data. Whereas for low-dimensional embeddings INNC provides on par or better performance as compared to the best (SRC or CRC) classifiers, for high-dimensional embeddings INNC gets behind the top SRC/CRC classifiers. This manifests itself more strongly when the data does not undergo a discriminant embedding as is the case for eigenfaces in Table 2, and when the dimensionality is high.

We test the scalability issue using the l_2-normalized raw grayscale pixels from bicubically downscaled images on the AR benchmark and report the results in Figure 8. This is an unfavorable case for INNC/INN0C since the data does not undergo a discriminant embedding. For large dimensionalities CRC and SRC benefit more than INNC and INN0C from the extra data available and outperform the latter in terms of recognition rates.

We can conclude that INN and INN0 come with a scalability issue with respect to the dimensionality of the data. It is not a strict scalability issue per se, since the performance does improve with the dimensionality of the data, but when compared with l_1-regularized representations the improvement rate is lower.

4.3. Classification results

4.3.1. Face and traffic sign recognition

The AR face and GTSRB traffic sign benchmarks are used to validate the power of our INN representation for the task of classification. Each image is given as a feature vector – here the grayscale pixels under different projections. The settings are those previously mentioned for dimensionality reduction. Table 2 shows the results on faces and Table 3 the results on traffic signs. On AR, a rather small dataset, the INN-based classifier provides the best results for low-dimensional features, while SRC takes the lead for medium to high-dimensional features, regardless the feature type. On the large traffic sign dataset INNC proves to be the best performer for the discriminative embeddings, while for Eigenfaces embeddings comes second to SRC with the Feature Sign solver, and this by a margin – 77.7% vs. 85.3%. Note that INNC is more than 10× faster

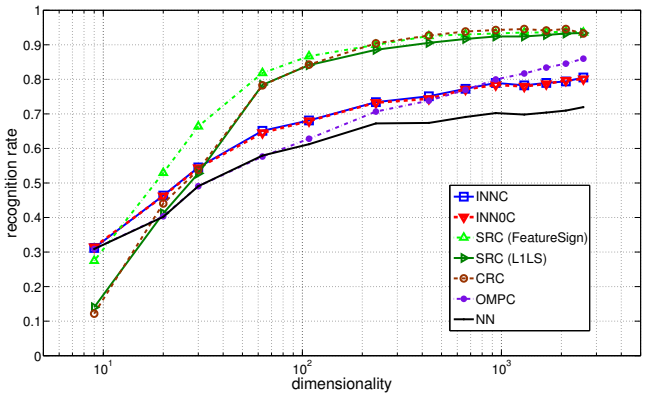


Figure 8: Recognition rates as a function of dimensionality on the AR dataset. The grayscale images where downscaled from 9 up to 2580 pixels, and l_2 normalized. The parameters were tuned for best overall performance. λ is set to 0.01 for INNC and INN0C, to 0.001 for FeatureSign, to 0.01 for L1LS in SRC, to 0.01 for CRC, and OMPC is used with $\epsilon = 0.1$.

Table 2: The face recognition results [%] of different methods on the AR database using different linear projections. We bold the top results and those in a 0.5 margin for each projection type and dimension.

	Dim	5	10	30	54	99	120	300
eigenfaces	INNC	**24.3**	**48.9**	69.3	74.4	77.4	77.7	79.7
	INN0C	**24.1**	**48.4**	68.8	74.1	77.1	77.7	79.4
	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**
	CRC	06.3	19.5	64.2	80.3	89.3	90.1	**93.8**
	OMPC	23.5	45.1	63.2	70.7	79.8	81.1	90.0
	NN	23.5	43.4	59.1	68.1	69.8	70.4	71.4
	LSVM	10.3	33.5	68.2	76.4	81.3	82.4	84.5
	Dim	5	10	30	54	99	120	300
LDA	INNC	**37.9**	**60.8**	**87.0**	89.4	92.4		
	INN0C	**37.5**	**60.8**	**86.6**	88.8	92.6		
	SRC(FeSg)	34.5	56.8	**87.1**	**92.1**	**94.4**		
	SRC(L1LS)	20.0	47.9	86.0	**92.4**	**94.9**		
	CRC	09.6	26.0	75.5	90.7	91.0		
	OMPC	37.2	58.8	80.0	85.0	91.6		
	NN	37.2	58.8	76.7	81.8	87.0		
	Dim	5	10	30	54	99	120	300
SRLP	INNC	**38.9**	**62.4**	85.1	89.3	92.9	93.0	93.0
	INN0C	**39.4**	**62.3**	85.3	88.6	92.4	93.0	93.1
	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**
	CRC	09.9	27.3	77.0	90.0	91.8	92.4	88.4
	OMPC	37.1	59.9	79.7	86.7	90.8	91.6	91.6
	NN	37.8	59.8	77.3	82.8	86.0	86.4	86.7
	Dim	5	10	30	54	99	120	300
INNLP	INNC	**38.8**	**62.1**	85.1	89.4	92.7	93.0	93.0
	INN0C	**38.5**	61.5	85.1	89.1	92.4	92.9	93.0
	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**
	CRC	09.7	27.6	76.3	89.6	91.7	92.6	88.0
	OMPC	37.1	60.1	79.1	87.1	90.8	91.6	91.0
	NN	37.3	59.7	77.4	82.7	86.0	86.1	86.9

than the SRC approaches. INN performs marginally better than INN0.

4.3.2. Scene recognition

To assess the recognition performance for scenes we use the Scene-15 benchmark [32] and follow the common experimental settings with 100 training images per class. We report in Table 4 the mean performance and standard deviation for 20 randomly generated train/test splits.

The NIMBLE features are extracted as in the original work [33], with 100 fixations per image and PCA-projected to a 500-dimensional embedding space, and augmented by normalized image coordinates. We aim for speed and NBSR uses the Feature Sign solver with $\lambda = 0.3$, while $\lambda = 0.1$ for INN in both the NBINN and INNSPM settings (see Section 3.2).

NBINN0 with NIMBLE features is marginally better than NBSR and both improve over NBNN. NBINN, however, is slightly worse than NBSR. The results are comparable with the ScSPM results using SIFT features [19]. A 10× speedup is achieved by NBINN over NBSR, rendering the approach comparable in running time to NBNN. The same happens for INNSPM over ScSPM, being comparable in run-time to the Locality-Constraint Linear Coding (LLC) [34] method. The INNSPM performance is on par with or better than that of the original ScSPM. It is definitely better than the learning-free NB approaches. It is noteworthy that INN now sees its performance decline when used in its refined version.

Table 3: The traffic sign classification accuracy[%] on the GTSRB benchmark. We use 300 eigenfaces, 42-dimensional LDA projections, 99-SRLP, 99-INNLP. We bold the top results and those in a 0.5 margin for each feature type.

classifier	eigenf.	SRLP$_{FeSg}$	INNLP	SRLP$_{Hmtp}$	LDA
INNC	77.72	**93.69**	**93.65**	**93.64**	**93.66**
INN0C	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
CRC	84.34	83.43	83.56	83.39	84.08
OMPC	76.17	91.92	91.83	91.80	91.59
NN	66.05	89.54	89.35	89.27	91.84
LSVM	81.22	87.87	87.92	87.38	87.00
IKSVM	**87.45**	89.51	89.06	89.66	86.30
POLYSVM	82.12	92.76	92.39	92.51	92.49
RBFSVM	81.34	92.43	92.57	92.28	92.46

Table 4: Performance comparison on Scene-15. We bold the top results.

Learned?	Train/test split Method	100/100 avg CR
yes	ScSPM+SIFT [19]	80.28 ± 0.93
yes	EMK+KDES [29]	87.5
yes	LScSPM+SIFT [30]	**89.75±0.5**
yes	NBNN&BoF kernels+SIFT [31]	85 ± 4
yes	NBNN-f_2+SIFT [31]	75 ± 3
yes	*INN*SPM+SIFT	80.7 ± 1
yes	*INN*0SPM+SIFT	81.4 ± 1
no	NB*INN*+NIMBLE	77.06 ± 1
no	NB*INN*0+NIMBLE	**78.23±1**
no	NB*NN*+NIMBLE	74.2 ± 1
no	NB*SR*+NIMBLE	77.75 ± 1

4.3.3. Image classification

We use the PASCAL VOC 2007 dataset [35] with the standard settings and report the results in Table 5 for the NBSR, NBINN and INNSPM methods as introduced in Section 3.2. NBINN outperforms the NBSR classifier both in speed and performance. NBINN achieves a mean average precision which is only 6% below the best entry of the PASCAL VOC 2007 challenge. The NB classifiers are used with the NIMBLE features as in the previous experiment. INNSPM with SIFT features performs better than ScSPM with SIFT features or LLC with HOG features and compares favorably with the LLC method in terms of its running time. In all of these experiments we have used INN's original formulation.

4.4. Discussion

We introduced Iterative Nearest Neighbors, a novel sparse representation. INN inherits the speed of k-NN approaches and the performance of sparse decompositions when solving an l_1-regularized least squares formulation.

We have shown:

(i) our INN to be on par or better with standard sparse recovery iterative algorithms, MP and OMP, under ideal as well as under noisy conditions,

(ii) our INN Linear Projection method for dimensionality reduction to be faster than the original SRLP,

(iii) our Naive Bayes INN for image classification to reach top results on the PASCAL VOC 2007 benchmark, while improving over NBNN and NBSR,

Table 5: Image classification results on PASCAL VOC 2007 benchmark

object class	aero	bicyc	bird	boat	bottle	bus	car	cat	chair	cow	
Best of VOC'07[35]	77.5	63.6	56.1	71.9	33.1	60.6	78.0	58.8	53.5	42.6	
LLC [34]	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	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[35]	54.9	45.8	77.5	64.0	85.9	36.3	44.7	50.9	79.2	53.2	59.4
LLC [34]	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	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**

(iv) our INN based Classifier to be faster and on par with the l_1-based SRC and the l_2-based CRC for face recognition (AR benchmark), and to be on par on traffic signs (GTSRB benchmark) with the best SRC and Feature Sign solver, but order(s) of magnitude faster, and

(v) our INNSPM pipeline for image classification to improve both the speed and performance over the original ScSPM [19], and to have similar speed and performance as Locality-Constraint Linear Coding (LLC) [34], one of the fastest and most robustly performing image classification frameworks, when run on the PASCAL VOC 2007 benchmark.

Moreover, INN has desirable properties, such as:

(i) steep convergence to an ϵ solution, determined by the exponentially decaying imposed weights,

(ii) fast nearest neighbor search as main computational step,

(iii) one regulatory parameter, that can be either the number of iterations or the residual weight (λ), depending on the application,

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

On the downside, as shown in Figures 7 and 8, INN is not able to reach the top performance of the l_1-regularized least squares representations for high-dimensional data.

5. Conclusions

This paper proposed a novel sparse representation, namely the Iterative Nearest Neighbors (INN). INN often succeeds in combining the power and complexity of l_1 or l_2-regularized least squares representations on the one hand, and the simplicity and speed of kNN methods on the other. INN is defined by an iterative algorithm. It compensates the residuals by iteratively picking the nearest neighbor to the working query. Depending on the application, the controlling parameter for INN is either the impact of the residual or the number of iterations. If one chooses the impact of the residual, the number of iterations is directly determined (thus, the maximum number of NN searches and selected samples) assuming measurement noise. Similarly, if one imposes the number of iterations, it is λ that follows.

INN is well-behaved, the smaller the parameter λ, the better INN's performance gets but the slower it is. In classification and dimensionality reduction applications INN provides a performance that is on par with CR and SR while being orders of magnitude faster. On the downside, INN benefits less than SR when the dimensionality of the data gets large.

Acknowledgments. The authors thank the reviewers for their useful comments. This work was partly supported by the European Commission FP7-231888-EUROPA project and the Flemish IWT/SBO ALAMIRE project.

References

[1] H. Zou, T. Hastie, R. Tibshirani, Sparse principal component analysis, Journal of Computational and Graphical Statistics 15(2) (2006) 262–286.

[2] X. He, P. Niyogi, Locality preserving projections, in: Proceedings of the Advances in Neural Information Processing Systems 16, NIPS (2003), MIT Press, Cambridge, MA, 2004.

[3] R. Timofte, L. Van Gool, Sparse representation based projections, in: Proceedings of the 22nd British machine vision conference - BMVC 2011, pp. 61.1–61.12.

[4] R. Fisher, The statistical utilization of multiple measurements, Annals of Eugenics 8 (1938) 376–386.

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

[6] S. Roweis, L. Saul, Nonlinear dimensionality reduction by locally linear embedding, in: Proceedings of IEEE International Conference on Computer Vision – ICCV 2001, volume 290, pp. 2323–2326.

[7] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in: Advances in Neural Information Processing Systems 14, MIT Press, 2001, pp. 585–591.

[8] B. Raducanu, F. Dornaika, A supervised non-linear dimensionality reduction approach for manifold learning, Pattern Recognition 45(6) (2012) 2432–2444.

[9] L. Goldfarb, What is distance and why do we need the metric model for pattern learning?, Pattern Recognition 25(4) (1992) 431–438.

[10] J. Wright, A. Y. Yang, A. Ganesh, S. Sastry, Y. Ma, Robust face recognition via sparse representation, IEEE Transactions on Pattern Analysis and Machine Intelligence 31(2) (2009) 210–227.

[11] L. Zhang, M. Yang, X. Feng, Sparse representation or collaborative representation: Which helps face recognition?, in: Proceedings of IEEE International Conference on Computer Vision, ICCV (2011), pp. 471–478.

[12] R. Timofte, L. J. Van Gool, Iterative nearest neighbors for classification and dimensionality reduction, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR (2012), pp. 2456–2463.

[13] S. Mallat, Z. Zhang, Matching pursuits with time-frequency dictionaries, IEEE Transactions on Signal Processing 41(12) (1993) 3397–3415.

[14] J. A. Tropp, A. C. Gilbert, Signal recovery from random measurements via orthogonal matching pursuit, IEEE Transactions on Information Theory 53(12) (2007) 4655–4666.

[15] Y. C. Pati, R. Rezaiifar, P. S. Krishnaprasad, Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition, in: Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems, and Computers (1993), pp. 40–44.

[16] V. Popovici, S. Bengio, J.-P. Thiran, Kernel matching pursuit for large datasets, Pattern Recognition 38(12) (2005) 2385 – 2390.

[17] O. Boiman, E. Shechtman, M. Irani, In defense of nearest-neighbor based image classification, in: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition - CVPR (2008), pp. 1–8.

[18] R. Timofte, T. Tuytelaars, L. J. Van Gool, Naive bayes image classification: Beyond nearest neighbors, in: K. M. Lee, Y. Matsushita, J. M. Rehg, Z. Hu (Eds.), Asian Conference on Computer Vision, ACCV (1), volume 7724 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 689–703.

[19] J. Yang, K. Yu, Y. Gong, T. S. Huang, Linear spatial pyramid matching using sparse coding for image classification, in: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR (2009), IEEE, 2009, pp. 1794–1801.

[20] A. Martinez, R. Benavente, The AR face database, Technical Report, CVC Tech. Report No. 24, 1998.

[21] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, D. Gorinevsky, An interior-point method for large-scale l_1-regularized least squares, IEEE Journal on Selected Topics in Signal Processing 1(4) (2007) 606–617.

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

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

[24] H. Lee, A. Battle, R. Raina, A. Y. Ng, Efficient sparse coding algorithms, in: B. Schölkopf, J. C. Platt, T. Hoffman (Eds.), Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, NIPS (2006), MIT Press, 2006, pp. 801–808.

[25] A. Yang, A. Ganesh, Z. Zhou, S. Sastry, 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.

[26] R. Timofte, L. J. Van Gool, Weighted collaborative representation and classification of images, in: Proceedings of the 21st International Conference on Pattern Recognition, ICPR (2012), IEEE, 2012, pp. 1606–1610.

[27] R. Timofte, L. Van Gool, Adaptive and weighted collaborative representations for image classification, Pattern Recognition Letters (2013) –.

[28] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, The German Traffic Sign Recognition Benchmark: A multi-class classification competition, in: Proceedings of International Joint Conference on Neural Networks, IJCNN (2011), IEEE, 2011, pp. 1453–1460.

[29] L. Bo, X. Ren, D. Fox, Kernel descriptors for visual recognition, in: Proceedings of the Advances in Neural Information Processing Systems, NIPS (2010), pp. 244–252.

[30] S. Gao, I. W.-H. Tsang, L.-T. Chia, P. Zhao, Local features are not lonely - laplacian sparse coding for image classification, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2010), pp. 3555–3561.

[31] T. Tuytelaars, M. Fritz, K. Saenko, T. Darrell, The NBNN kernel, in: D. N. Metaxas, L. Quan, A. Sanfeliu, L. J. V. Gool (Eds.), Proceedings of IEEE International Conference on Computer Vision, ICCV (2011), IEEE, 2011, pp. 1824–1831.

[32] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR (2) (2006), IEEE Computer Society, 2006, pp. 2169–2178.

[33] C. Kanan, G. W. Cottrell, Robust classification of objects, faces, and flowers using natural image statistics, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2010), pp. 2472–2479.

[34] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, Y. Gong, Locality-constrained linear coding for image classification, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2010), pp. 3360–3367.

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