Nature-Inspired Graph Optimization for Dimensionality Reduction

Murillo G. Carneiro*, Thiago H. Cupertino†, Ran Cheng‡, Yaochu Jin§ and Liang Zhao¶
*Faculty of Computing, Federal University of Uberlândia, Uberlândia, Brazil, 38400–902
Email: mgcarneiro@ufu.br
†Department of Information, São Paulo State Finance Secretariat, São Paulo, Brazil, 01017–911
‡School of Computer Science, University of Birmingham, Birmingham, United Kingdom, B15 2TT
§Department of Computer Science, University of Surrey, Guildford, United Kingdom, GU2 7XH
¶Department of Computing and Mathematics, University of São Paulo, Ribeirão Preto, Brazil, 14040–901

***Abstract*—Graph-based dimensionality reduction has attracted a lot of attention in recent years. Such methods aim to exploit the graph representation in order to catch some structural information hidden in data. They usually consist of two steps: graph construction and projection. Although graph construction is crucial to the performance, most research work in the literature has focused on the development of heuristics and models to the projection step, and only very recently, attention was paid to network construction. In this work, graph construction is considered in the context of supervised dimensionality reduction. To be specific, using a nature-inspired optimization framework, this work investigates if an optimized graph is able to provide better projections than well-known general-purpose methods. The proposed method is compared with widely used graph construction methods on a range of real-world image classification problems. Results show that the optimization framework has achieved considerable dimensionality reduction rates as well as good predictive performance.**

***Keywords*—Graph-based dimensionality reduction; nature-inspired graph optimization; graph-based machine learning;**

I. Introduction

Dimensionality reduction (DR) is a well-known machine learning and pattern recognition task which aims to reduce the dimension of the input data while maintaining the features of interest [1]. Among various DR methods, the graph-based ones have attracted increasing interest [2]. Many of these methods fall into a graph embedding framework [3] commonly consisting of two steps. The first step is graph construction, in which data items are taken as nodes and their edges are usually given by some affinity measure. The second step learns a projection which transforms the original high-dimensional data into a lower-dimensional space.

Derived from ISOMAP [4], Locally Linear Embedding [5], Laplacian eigenmap [6], Marginal Fisher Analysis (MFA) [3], t-SNE [7] and others, a wide range of projection models and heuristics have been proposed in the literature. By contrast, graph construction, which is considered as an important step in graph-based DR, has received attention only recently [8]–[10], especially in the supervised DR context. In fact, the most commonly used graph construction methods are based on the k nearest neighbors (kNN) criterion [11]. Such a neighborhood graph does not necessarily benefit subsequent DR task as it is artificially defined in advance. In this sense, GoLPP [12] and GoDRSC [13] aimed to integrate graph construction with a specific DR process into a unified framework, thereby obtaining an optimized graph rather than a predefined one. Despite both related works were designed for unsupervised DR, they also serve as a motivation to create new graph construction approaches for supervised DR.

This article investigates a nature-inspired optimization framework for graph-based supervised DR. The framework, which is based on [14], is designed to build up the graph while optimizing a given quality function. The quality function used is based on the MFA approach presented in [3] and combines locality and class label information to represent intra-class compactness and interclass separability. In contrast to existing works where a specific value is assigned to the connections by all vertices, the connections here are iteratively updated by a robust particle swarm optimization method [15]. Experiments have been performed over a range of real-world image classification problems. In addition, the proposed graph construction approach is compared with two general-purpose methods widely used in the literature, symmetric and mutual kNN [11].

The remainder of the work is organized as follows: Section II covers a relevant background about DR problem, graph construction methods and the graph embedding approach considered here; Sections III and IV describes the proposed approach and the experimental results, respectively; and Section V concludes the article.

II. Background

A. Problem Definition

In the problem concerned here, the algorithms receive as input a training data set denoted by $X_{Train} = \{\mathbf{x}_i,\ i = 1, \ldots, n\}$, containing n labeled items, and a test data set $X_{Test} = \{\mathbf{x}_i,\ i = n+1, \ldots, m\}$, containing m unlabeled items. Each item is described by a attributes, that is, a vector $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{ia}]^T$, and belongs to a single class $l_i \in \{1, \ldots, C\}$, where C is the number of classes. The goal of the proposed technique is to perform dimensionality reduction by using the information provided by the labeled data set X_{Train} in order to improve classification accuracy or, at least, to speed up the classification process of the unlabeled data set

X_{Test} without decreasing the accuracy, provided that a small number a' of the projected attributes is used ($a' < a$).

B. Graph Construction Methods

Consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each node $v_i \in \mathcal{V}$ represents a data item $\mathbf{x}_i \in X_{Train}$. Let kNN(i) be the set of k nearest neighbors of $\mathbf{x}_i$ calculated from a distance matrix $\mathbf{S}$. The adjacency matrix $\mathbf{A}$ of a kNN-graph is obtained as follows:

$$\mathbf{A}_{ij} = \begin{cases} 1, & \text{if } \mathbf{x}_j \in k\text{NN}(i) \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

As the kNN-graph may not be symmetric, two strategies are commonly used to assure this: symmetric kNN and mutual kNN. A symmetric kNN graph (SkNN) is obtained as follows:

$$\mathbf{A} = \max(\mathbf{A}, \mathbf{A}^T) \ . \quad (2)$$

By contrast, the mutual kNN graph (MkNN) is given by:

$$\mathbf{A} = \min(\mathbf{A}, \mathbf{A}^T) \ . \quad (3)$$

C. Graph Embedding Approach

In this article, the projection step is conducted by the Marginal Fisher Analysis criterion [3]. This method characterizes intra-class compactness and inter-class separability by using the preserving (or intrinsic) graph and the penalty graph, respectively. To be specific, MFA seeks w which minimizes the intra-class compactness and maximizes the inter-class separability at the same time, i.e.:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{\mathbf{w}^T X L X^T \mathbf{w}}{\mathbf{w}^T X L^P X^T \mathbf{w}}, \quad (4)$$

which can be solved by the generalized eigenvalue problem by using the equation $XLX^T\mathbf{w} = \lambda XL^PX^T\mathbf{w}$. In this formulation, L denotes the Laplacian matrix of the preserving network A, which can be found via the following operation:

$$L = D - A, \ D_{ii} = \sum_{i \neq j} A_{ij}, \forall i. \quad (5)$$

Conversely, the constraint matrix B can be viewed as the adjacency matrix of a penalty network A^P, so that $B = L^P = D^P - A^P$. The penalty network conveys information about which vertices should not be linked together, that is, which instances should be far apart after the dimensionality reduction process. The similarity preservation property from the graph-preserving criterion has a two-fold explanation. For larger similarity between samples $\mathbf{x}_i$ and $\mathbf{x}_j$, the distance between $\mathbf{y}_i$ and $\mathbf{y}_j$ should be smaller to minimize the objective function. Likewise, smaller similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$ should lead to larger distances between $\mathbf{y}_i$ and $\mathbf{y}_j$ for minimization [3].

III. Model Description

In this section, the nature-inspired optimization framework (NIO for short hereafter) is described in details. Unlike most graph embedding DR techniques, which focuses on the projection step, the proposed method focuses on the graph construction step, i.e., it assumes that by providing an optimal network regarding a given processing goal, the general performance can be improved.

A. Overview

The graph optimization framework employed in this study is based on that presented in [14], which conducts graph structural optimization using the recently proposed social learning particle swarm optimization (SL-PSO) [15]. In a few words, SL-PSO initializes a swarm of particles, with each one denoting a randomly initialized decision vector. Unlike traditional particle swarm optimization algorithms, it does not memorize the historical best positions. Instead, it sorts the swarm according to the fitness values of the particles, and as a consequence, each particle is made to learn from any better particles in the current swarm. Such salient features make SL-PSO robust on high-dimensional problems and a similar version of SL-PSO has been successfully applied to high-dimensional feature selection [16].

The main steps of the general framework adopted here are illustrated by Fig. 1. Initially, SL-PSO creates a population of particles, where each particle P_i is composed by two graphs $\mathcal{G}$ and B. Then, at each iteration t, the particles are evaluated and updated (Δ) according to a quality function f. At the end, SL-PSO returns the particle with the best quality value, which contains the graphs to be equipped into the DR approach. To be specific, NIO is divided in two phases:

- **Optimization** In this phase, NIO employs SL-PSO algorithm to construct the preserving and penalty graphs from the training data X_{Train} according to the optimization of a given quality function f under a given validation data set X_{Valid}.
- **Testing** Here, NIO is equipped with the preserving and penalty graphs of the best particle learned during the optimization phase. The projection attributes learned from these graphs are then employed to reduce the number of features as well as to classify any unlabeled data $\mathbf{x}_i \in X_{test}$.

B. Optimization Phase

The optimization phase consists of three main concepts: network representation, mapping heuristic, and preserving and penalty graph. Network representation refers to model particles into networks and vice versa. Due to the high number of possible particle (network) configurations, two mapping heuristics are designed in order to reduce the time complexity. Based on such mapping heuristics, the preserving and penalty graphs convey information about intra-class and inter-class vertices, respectively. Following we discuss each of these concepts.

1) Network Representation: Given a swarm of m particles $\mathbf{P} = \{P_1, \ldots, P_m\}$, each particle $P_i \in \mathbf{P}$ can be represented as follows:

$$P_i = \{\mathcal{G}_i, B_i\}, \quad (6)$$

where $\mathcal{G}_i$ and B_i denote the preserving and penalty graph, respectively.

In the data representation designed for NIO, each particle P_i denotes a labeled data item $\mathbf{x}_j \in X_{Train}$ as a vertex $v_j \in \mathcal{V}_i$, i.e.:

$$\mathcal{V}_i = \{v_1, \ldots, v_n\}, \quad (7)$$

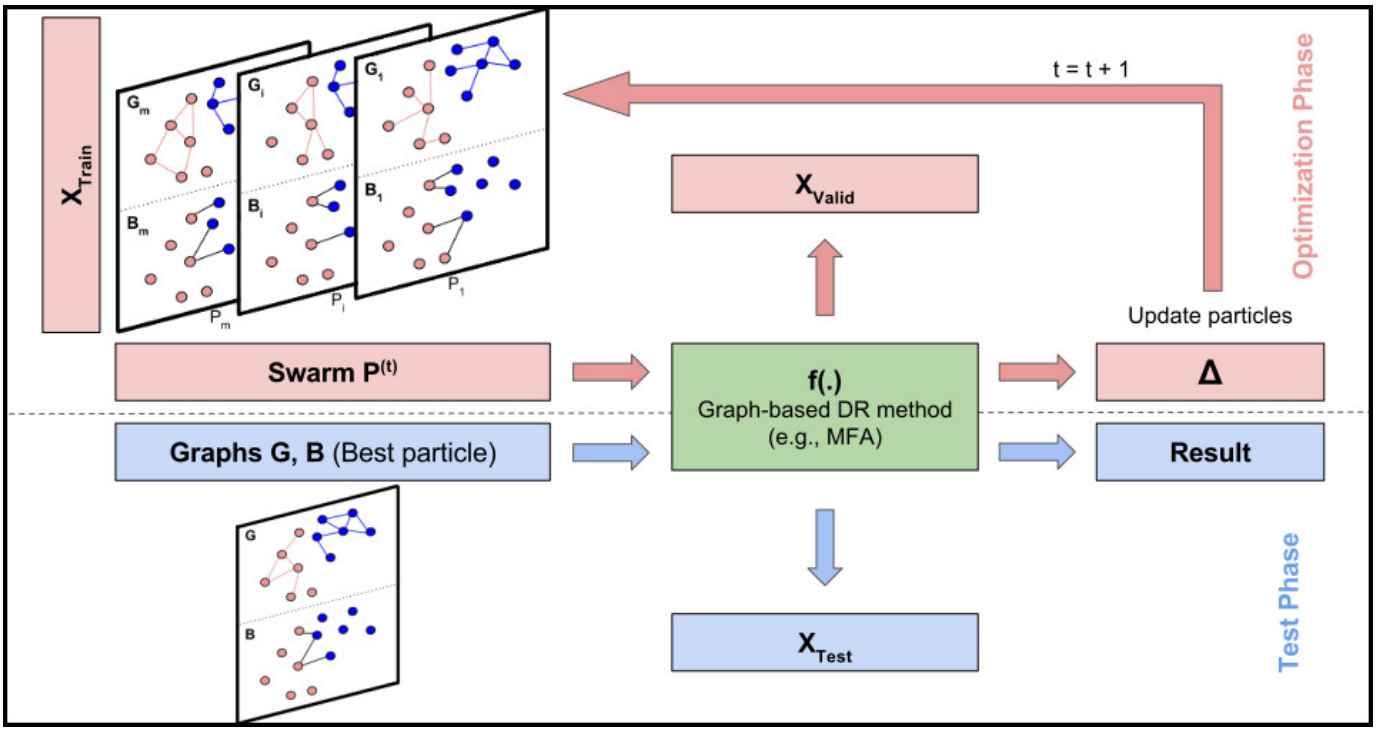


Fig. 1. A general view of the nature-inspired graph optimization framework for dimensionality reduction.

where n means the number of data items (or vertices). The possible connections of each vertex v_j are represented by:

$$v_j = \{p_{j1}, \ldots, p_{jq}\}, \tag{8}$$

with q denoting the maximum number of links and p_{jk} denoting the probability of a link between node v_j and node v_k. Values of p_{jk} are continuous and vary between $[0, 1]$ in order to be manipulated by SL-PSO. By contrast, the graph connections are encoded with binary values, denoted as p'_{jk}, which can be obtained as:

$$p'_{jk} = \begin{cases} 1, & \text{if } p_{jk} \geq 0.5, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

2) Mapping Heuristic: Given a graph with n vertices, the total number of possible edges is n^2 and the complexity of the search space is $\mathcal{O}(n^2)$. However, since n can be as large as hundreds or even thousands, a search complexity of $\mathcal{O}(n^2)$ is infeasibly expensive. To address this issue, we designed a mapping function that creates a sub-dimensional space based on the features of the given data set X_{Train}, which reduces the search complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ ($\mathcal{O}(n \cdot q), q \ll n$) by using the following steps:

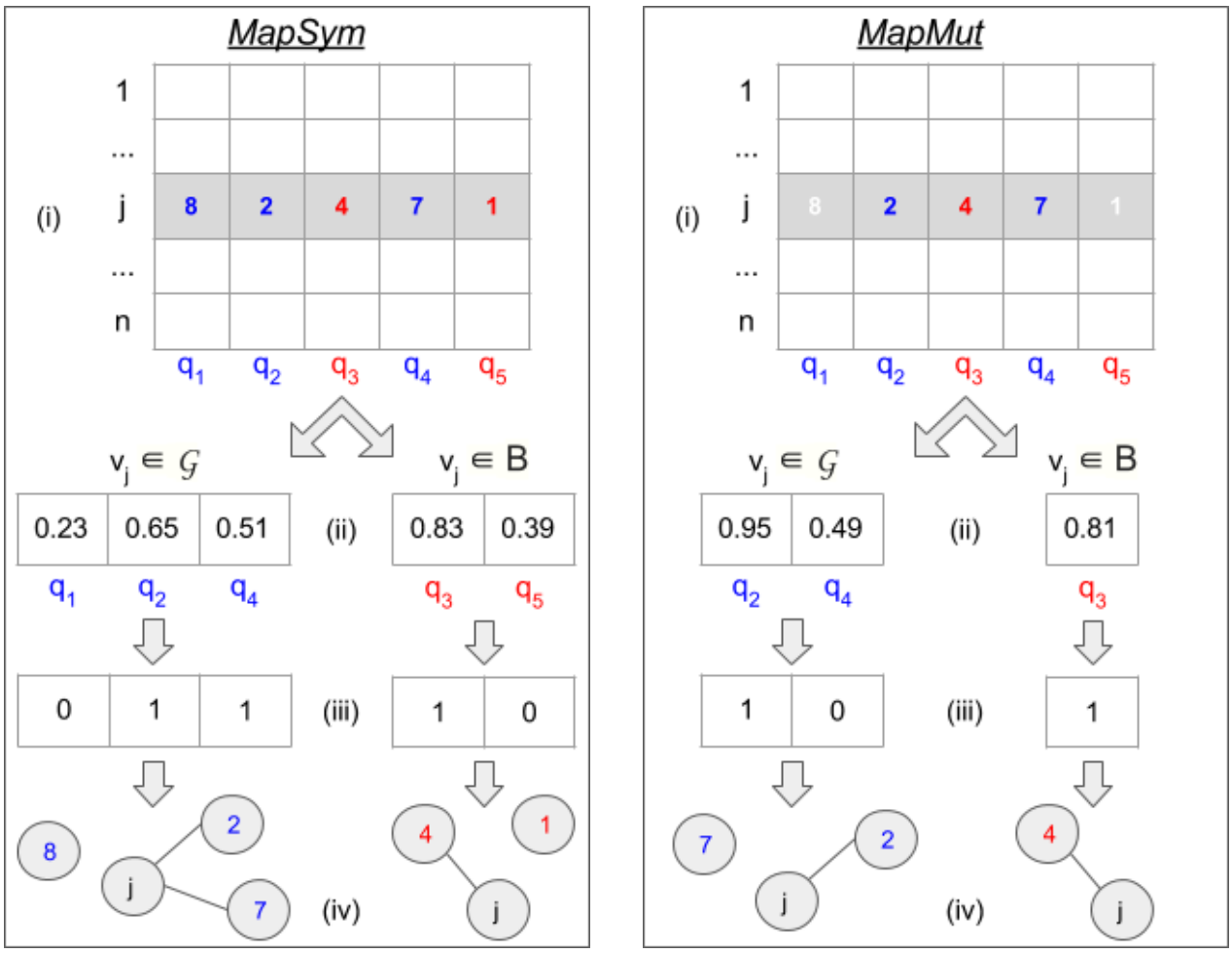
1) Compute the similarity among the instances;
2) Select q more similar vertices for each vertex v_i;
3) Create $\text{Map}_{n \times q}$ matrix according to some mapping heuristic.

Two mapping heuristics for DR are proposed in this article: symmetric *MapSym* and mutual *MapMut*. Given two vertices v_j and v_z (with $z = Map_{jk}$), *MapSym* is based on SkNN and considers a possible link e_{jz} if v_z is among the q nearest neighbors of v_j or vice versa. By contrast, *MapMut*, which is based on MkNN, assumes a possible connection between e_{jz} only if both vertices are among the q nearest neighbors of each one, i.e., $v_j \in k\text{NN}(v_z)$ and $v_z \in k\text{NN}(v_j)$.

3) Preserving and Penalty Graphs: After the mapping heuristic is calculated, the edges of both graphs $\mathcal{G} = \{\mathcal{V}, \mathcal{E}^{\mathcal{G}}\}$ and $B = \{\mathcal{V}, \mathcal{E}^{B}\}$ can be obtained. Let us consider a connection e_{jz}. To be specific, such a link will belong to the preserving graph $\mathcal{G}$ if both data items $\mathbf{x}_j$ and $\mathbf{x}_z$ have the same class label; otherwise, it will be represented in the penalty graph B; i.e.:

$$e_{jz} \in \begin{cases} \mathcal{E}^{\mathcal{G}}, & \text{if } l_j = l_z, \\ \mathcal{E}^{\mathcal{B}}, & \text{otherwise.} \end{cases} \tag{10}$$

An illustrative example covering the network representation, the mapping heuristic and the preserving and penalty graphs is shown by Figure 2. The step (i) in the figure presents both mapping heuristics investigated: *MapSym* (Fig. 2a) and *MapMut* (Fig. 2b). It also emphasizes particular characteristics of each one. For example, vertices in positions q_1 and q_5 in Fig. 2b do not have links with v_j as they are not mutual nearest neighbors (i.e., v_j is not among the q nearest neighbors of them). Steps (ii) and (iii) show an example about the transformation of the solutions from the vector-based probability, which is manipulated by SL-PSO, to the graph. Step (iv) denotes the formation of the preserving graph (left-hand side) and the penalty graph (right-hand side).



(a) *MapSym*: Symmetric Mapping Heuristic

(b) *MapMut*: Mutual Mapping Heuristic

Fig. 2. Illustrative example of the nature-inspired framework representation considering *MapSym* and *MapMut* mapping heuristics. Steps (i-iv) denote the mapping matrix, the SL-PSO representation, the graph encoding and the plotted graph. If vertices have the same class label as v_j, they are colored by blue; otherwise, by red color. In Fig. 2b, the white color denotes vertices that do not are mutual nearest neighbors of v_j.

C. Test Phase

The test phase consists of applying the best particle (i.e., preserving and penalty graphs) learned from the optimization phase to any new test data $\mathbf{x}_i \in X_{test}$, which means that each unlabeled data is projected to another dimensional space and classified according to the quality function f.

D. Quality Function

The quality function f evaluates each particle (or network configuration) in NIO. Basically, f consists of a graph embedding DR technique and a classifier. The former receives the preserving and penalty graphs $\mathcal{G}$ and B, respectively, and calculates the projection vector after the Laplacian matrix of both graphs is obtained, such as defined in (4) and (5). The latter is responsible to evaluate the projected data in terms of prediction and reduction of features. In terms of predictive performance, a classifier predicts the labels of the projected validation data after being trained over the projected training data. In terms of dimensionality reduction, the predictive performance of such a classifier is evaluated over a distinct number of features.

Formally, let Q be a vector in which each position Q_i denotes a given number of projected features to be considered, let also $Acc^{(Q_i)}$ be the predictive accuracy achieved by a classifier after training and predicting over Q_i projected features. While $\max(Acc)$ is returned in the test phase, the optimization phase returns the averaged accuracy among the number of the projected features, i.e.:

$$f = \frac{1}{|Q|} \sum_{i} Acc^{(Q_i)}. \tag{11}$$

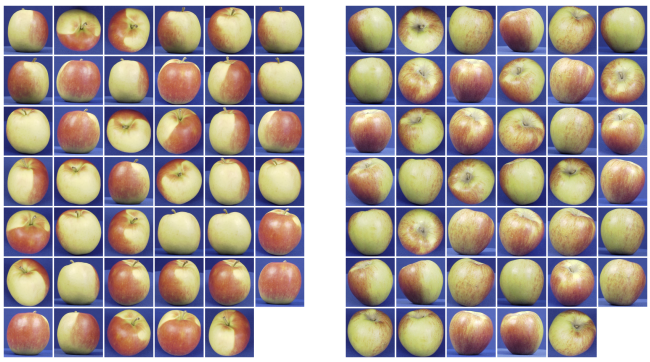
Briefly, each particle in SL-PSO is evaluated by taking into account its predictive performance under $|Q|$ cases, each one denoting a distinct number of projected features. Thus, (11) provides a smooth function to conduct the optimization phase.

IV. Experimental Results

Experiments have been conducted in order to compare the proposed method against widely used graph construction methods, such as SkNN and MkNN, where the performance is evaluated over high-dimensional real-world data sets in terms of reduction of the number of features and predictive accuracy.

The data sets are from the ETH-80 collection, which comprises a total of 3280 images divided in 8 categories: Apple, Car, Cow, Cup, Dog, Horse, Pear and Tomato, as shown by Fig. 3. Each category contains 10 objects that span large in-class variations while still clearly belonging to the category. For each object, there are 41 images from viewpoints spaced equally over the upper viewing hemisphere (at distances from 22.5^o to 26.0^o). For instance, Fig. 4 shows the 41 images of two apples objects from Apple category.

Fig. 3. The eighty objects of the ETH-80 collection.



(a) Apple category - Object 2 (b) Apple category - Object 7

Fig. 4. Two objects of the Apple category in ETH-80 collection.

The preprocessing of the data sets included the following steps: the images were down-sampled from 128×128 (original size) to 32×32 to speed up processing; a total of 512 features were extracted for each image by calculating its histogram; and the similarities among images are calculated by the euclidean distance.

Each simulation consists of a 10-fold stratified cross-validation process. In this process, the data set is split in 10 disjoint sets and, in each run, 9 sets are used as training data and 1 set is used as the test data, resulting in a total of 10 runs. In each run, the training data is divided: 75% as sub-training (X_{Train}) and 25% as validation (X_{Valid}). By doing this, we assure an unbiased learning as the testing data is outside of the learning process. After the dimensionality reduction step, the projected data set was classified by using the nearest-neighbor classification rule. The results of each run are averaged over 5 different executions (using different random seed).

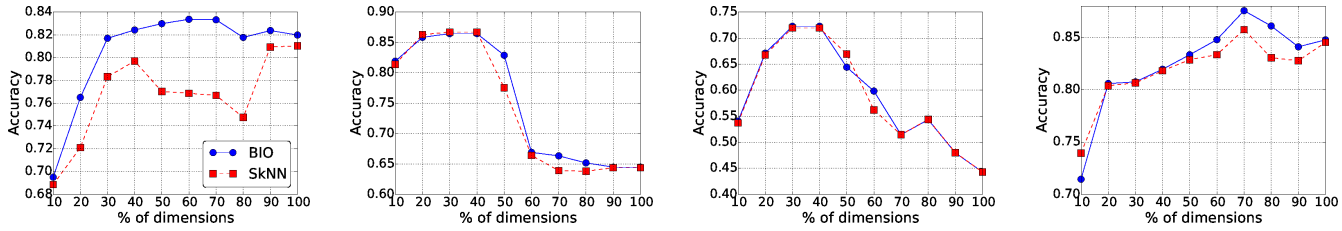
The optimization framework has four parameters to be set: the size of the swarm population m, the number of iterations it, the maximum number of possible links q and the number of projected features to be considered by the quality function Q, which are defined as $m = 100$, $it = 100$, $q = 3$ and $Q = \{10\%, 20\%, \ldots, 100\%\}$ of the original number of features.

The experiments are divided in two groups in order to better evaluate the proposed mapping heuristics. Firstly, NIO is equipped with *MapSym* heuristic and its predictive performance is compared with the widely used SkNN method. Table I lists the best average accuracy obtained by each approach. The table also includes results of the 1NN classifier considering all the original features. In order to analyze the results, the Wilcoxon's test is performed for pairwise comparison of the algorithms [17]. The statistical test results, presented in the same table, reveal that the graph embedding DR technique performs statistically better with our graph optimization framework than SkNN (with a confidence level of 90%) and the baseline (95%). Figure 5 presents a comparative analysis between both graph-based DR approaches on the first four data sets, which demonstrates NIO has much better predictive performance on the images of Apple category.

TABLE I
Results in terms of average accuracy considering a baseline over the original features (1NN) and a graph embedding DR method in which the graphs are provided by a widely used method (SkNN) or by the optimization framework equipped with the proposed symmetric mapping heuristic (NIO-*MapSym*). The last line of the table presents results of the Wilcoxon statistical test.

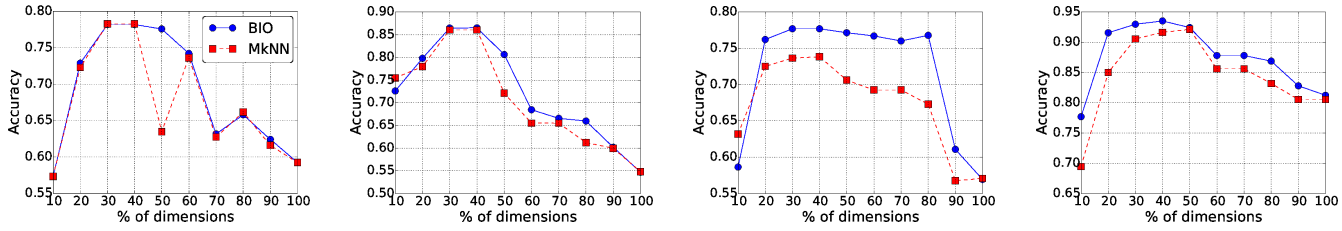
Dataset	Orig. Features	Graph Embedding DR	
	1NN	SkNN	NIO-*MapSym*
Apple	85.57 ± 2.56	81.03 ± 1.21	83.37 ± 1.85
Car	75.82 ± 4.31	86.71 ± 0.87	86.46 ± 0.96
Cow	49.94 ± 4.40	71.93 ± 1.76	72.28 ± 1.70
Cup	89.14 ± 2.26	85.74 ± 1.12	87.56 ± 1.86
Dog	63.13 ± 3.95	78.89 ± 1.72	78.37 ± 1.54
Horse	58.72 ± 4.29	86.47 ± 0.40	86.88 ± 0.43
Pear	59.99 ± 3.69	72.79 ± 1.96	77.04 ± 2.01
Tomato	88.12 ± 3.85	91.49 ± 1.03	92.83 ± 1.42
NIO-*MapSym*	$p < 0.05$	$p < 0.10$	$\sim$

Now we move on to evaluate NIO using the mutual mapping heuristic. In this group of simulation, the optimization



(a) Apple category (b) Car category (c) Cow category (d) Cup category

Fig. 5. Comparative analysis between NIO-*MapSym* and SkNN over ETH-80 categories in function of average accuracy and number of projected features.



(a) Dog category (b) Horse category (c) Pear category (d) Tomato category

Fig. 6. Comparative analysis between NIO-*MapMut* and MkNN over ETH-80 categories in function of average accuracy and number of projected features.

framework is compared against MkNN, a well-known graph formation method. Table II presents the average accuracy of both graph embedding DR approaches. 1NN results are the same of the Table I. Again, the results of the Wilcoxon test show that NIO-*MapMut* graph embedding DR approach performs statistically better than the SkNN one (90%). A comparative analysis between both methods is presented in Fig. 5. The figure shows NIO method performs better than MkNN in most scenarios, especially those considering Pear and Tomato categories.

TABLE II
AVERAGE ACCURACY OF THE 1NN CLASSIFIER OVER THE ORIGINAL FEATURES (1NN) AND OF THE GRAPH EMBEDDING DR EQUIPPED WITH THE WELL-KNOWN MkNN GRAPH CONSTRUCTION METHOD AND WITH THE BIOINSPIRED OPTIMIZATION FRAMEWORK USING THE MUTUAL MAPPING HEURISTIC (NIO-*MapMut*). LAST LINE PROVIDES RESULTS OF THE WILCOXON STATISTICAL TEST.

Dataset	Orig. Features	Graph Embedding DR	
	1NN	MkNN	NIO-*MapMut*
Apple	85.57 ± 2.56	81.03 ± 1.21	83.81 ± 1.41
Car	75.82 ± 4.31	86.91 ± 0.96	86.17 ± 0.90
Cow	49.94 ± 4.40	72.03 ± 1.67	72.08 ± 1.67
Cup	89.14 ± 2.26	85.79 ± 1.17	88.03 ± 1.74
Dog	63.13 ± 3.95	78.29 ± 1.62	78.19 ± 1.03
Horse	58.72 ± 4.29	86.07 ± 0.85	86.52 ± 0.26
Pear	59.99 ± 3.69	73.84 ± 1.70	77.69 ± 1.28
Tomato	88.12 ± 3.85	92.13 ± 1.06	93.52 ± 0.88
NIO-*MapMut*	$p < 0.05$	$p < 0.10$	∼

Finally, Fig. 7 presents a visual comparison between the original features space and that projected by our method. One can see the proposed NIO method has achieved reasonable dimension reduction rates, between 30% and 70% of the original features.

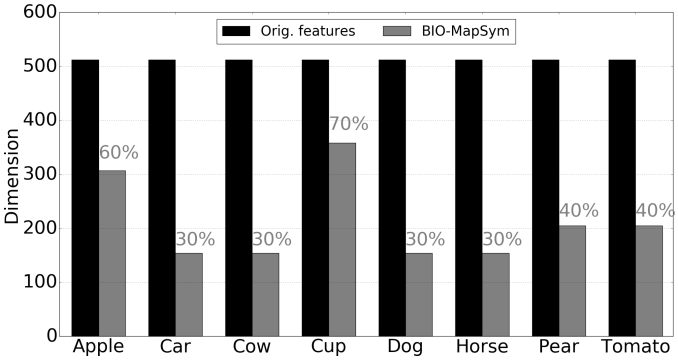


Fig. 7. Analysis of the reduction of the number of features in comparison with the original features space.

V. CONCLUSION

This article investigated graph construction for dimensionality reduction by using a nature-inspired optimization framework. Different from most work in the literature, where a specific value defines the number of connections for all vertices, our framework is able to build up the preserving and penalty graphs while optimizing the performance of the graph embedding approach, i.e., it integrates graph construction and projection into a unified framework. Experiments over a range of real-world image classification problems have shown the advantages over some other widely used graph construction methods. Moreover, results show that the optimization framework achieved considerable dimension reduction rates while also achieving good predictive performance.

Future work includes the investigation of embedding other graph-based DR approaches into our optimization framework; the comparison with other optimization and graph construction methods; and the evaluation of other quality functions and real-world problems.

ACKNOWLEDGMENT

Authors thank the financial support given by the São Paulo State Research Foundation - FAPESP (grants numbers 2012/07926-3, 2011/50151-0 and 2013/07375-0). Authors also acknowledge support from the Brazilian Coordination for the Improvement of Higher Education - CAPES and the Brazilian National Council for Scientific and Technological Development - CNPq.

REFERENCES

[1] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley, 2007.

[2] P. Foggia, G. Percannella, and M. Vento, "Graph matching and learning in pattern recognition in the last 10 years," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 28, no. 01, p. 1450001, 2014.

[3] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 40 –51, 2007.

[4] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[5] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[6] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[7] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[8] X. Zhu, "Semi-supervised learning literature survey," Tech. Rep., 2008.

[9] T. H. Cupertino, M. G. Carneiro, and L. Zhao, "Dimensionality reduction with the k-associated optimal graph applied to image classification," in *IEEE International Conference on Imaging Systems and Techniques*, 2013, pp. 366–371.

[10] M. G. Carneiro, T. H. Cupertino, and L. Zhao, "K-associated optimal network for graph embedding dimensionality reduction," in *IEEE International Joint Conference on Neural Networks*, 2014, pp. 1660–1666.

[11] K. Ozaki, M. Shimbo, M. Komachi, and Y. Matsumoto, "Using the mutual k-nearest neighbor graphs for semi-supervised classification of natural language data," in *ACL Conference on Computational Natural Language Learning*, 2011, pp. 154–162.

[12] L. Zhang, L. Qiao, and S. Chen, "Graph-optimized locality preserving projections," *Pattern Recognition*, vol. 43, no. 6, pp. 1993–2002, 2010.

[13] L. Zhang, S. Chen, and L. Qiao, "Graph optimization for dimensionality reduction with sparsity constraints," *Pattern Recognition*, vol. 45, no. 3, pp. 1205–1210, 2012.

[14] M. G. Carneiro, L. Zhao, R. Cheng, and Y. Jin, "Network structural optimization based on swarm intelligence for highlevel classification," in *IEEE International Joint Conference on Neural Networks*, 2016, pp. 3737–3744.

[15] R. Cheng and Y. Jin, "A social learning particle swarm optimization algorithm for scalable optimization," *Information Sciences*, vol. 291, pp. 43–60, 2015.

[16] S. Gu, R. Cheng, and Y. Jin, "Feature selection for high-dimensional classification using a competitive swarm optimizer," *Soft Computing*, pp. 1–12, 2016.

[17] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.