Int. J. Appl. Math. Comput. Sci., 2017, Vol. 27, No. 1, 169–180
DOI: 10.1515/amcs-2017-0012

DIMENSION REDUCTION FOR OBJECTS COMPOSED OF VECTOR SETS

MARTON SZEMENYEI [a,*], FERENC VAJDA [a]

[a]Department of Control Engineering and Information Technology
Budapest University of Technology and Economics, Magyar tudosok krt. 2, 1117, Budapest, Hungary
e-mail: {szemenyei,vajda}@iit.bme.hu

Dimension reduction and feature selection are fundamental tools for machine learning and data mining. Most existing methods, however, assume that objects are represented by a single vectorial descriptor. In reality, some description methods assign unordered sets or graphs of vectors to a single object, where each vector is assumed to have the same number of dimensions, but is drawn from a different probability distribution. Moreover, some applications (such as pose estimation) may require the recognition of individual vectors (nodes) of an object. In such cases it is essential that the nodes within a single object remain distinguishable after dimension reduction. In this paper we propose new discriminant analysis methods that are able to satisfy two criteria at the same time: separating between classes and between the nodes of an object instance. We analyze and evaluate our methods on several different synthetic and real-world datasets.

Keywords: dimension reduction, discriminant analysis, object recognition, registration.

1. Introduction

Dimension reduction procedures are found in numerous applications in the areas of object classification (Yasuoka *et al*., 2004; Surendiran and Vadivel, 2011; Liu *et al*., 2008), clustering (Boutsidis *et al*., 2011), data exploration (Kulczycki and Łukasik, 2014), feature selection (Song *et al*., 2010) or feature extraction (Lin, 1992). The classical methods such as principal component analysis (PCA) (Jolliffe, 2002), independent component analysis (ICA) (Hyvärinen *et al*., 2001) and linear discriminant analysis (LDA) (McLachlan, 2004) are most commonly employed. Unsupervised methods such as fuzzy k-means (FKM) have also shown great results in the context of text retrieval (Kumar, 2009).

In certain applications (Szemenyei and Vajda, 2015), however, it is more appropriate to describe instances of objects as unordered sets or graphs of feature vectors. We will refer to these classes as structured composite classes (SCCs). The individual feature vectors of an instance will be referred to as *nodes* of the object or class. While it is possible to apply typical dimension reduction methods to this problem, these algorithms will not produce optimal results when applied to SCCs.

SSCs appear in a wide range of computer vision applications. It is common to describe the visual appearance of an object by using local image features, such as SIFT (Lowe, 2004). Some models, such as the bag of visual words (Fei-Fei and Perona, 2005), describe objects as sets of visual features, while others, such as part-based or constellation models (Felzenszwalb *et al*., 2010) treat them as graphs. The local features approach also appears in 3D shape recognition (Bronstein *et al*., 2010). Another occurrence of SSCs is in the works of Schnabel *et al*. (2008) or Szemenyei and Vajda (2015), who describe 3D shapes as graphs of primitive shapes. It is worth noting that graphs of vectors can be reduced to sets of vectors by embedding graph nodes in a vector space, as is done by Demirci *et al*. (2011).

Some applications (Schnabel *et al*., 2008; Szemenyei and Vajda, 2015; Demirci *et al*., 2011) involve spotting or localizing subgraphs or subsets corresponding to a certain object class (segmentation by recognition). In these cases it is a straightforward strategy to recognize individual nodes and use node labels to infer the presence of an object from a given class. These methods do not only require discrimination between classes in general but also between the individual nodes of the objects. Being able to tell different nodes apart also provides the opportunity to weigh them using their relative importance.

This extra criterion is extremely useful in cases

*Corresponding author

where object registration or pose estimation is also needed. If it is easy to discriminate between nodes, then it is possible to learn a class model consisting of multiple nodes (including the relative position of the individual nodes). Using node-by-node matching pose estimation can be performed. However, if the nodes are too similar after dimension reduction, the matching will be ambiguous. It is worth noting that in most of these applications only class labels are available, that is, individual nodes are not labeled based on their similarity.

In this paper we present dimension reduction methods for structured composite classes by extending existing discriminant analysis algorithms. Our methods introduce the within-instance scatter matrix to ensure the separation between nodes of a given class. We also examine the relevant properties of the within-instance scatter matrix. Furthermore, we provide a method to use the labeling information of SCCs to avoid the clustering step that is required by some methods.

In the next section we give a brief overview of dimension reduction methods, especially linear discriminant analysis, and present some of the more recent results which are applicable to SCCs. In Sections 3 and 4 we present our methods for performing discriminant analysis on those that satisfy both of the aforementioned criteria. Lastly, in Section 5 we demonstrate the efficiency of our method by evaluating it on synthetic and real-world datasets and comparing its performance against typical algorithms.

2. Dimension reduction

In this section we give a brief overview of dimension reduction methods with the emphasis on linear discriminant analysis and its variants. The second part of the section is devoted to mixture and subclass extensions of LDA.

One of the most well-known dimension reduction methods is PCA (Jolliffe, 2002), which finds the linear combinations of the original dimensions that maximize the variance of the dataset. Thus, it computes the optimal compression of the dataset. Since PCA does not require class labels, it is widely used in unsupervised cases. However, for classification problems PCA might not select the dimensions that are most useful for telling different classes (Yang and Yuan, 2009).

Discriminant analysis (DA) techniques, on the other hand, are supervised procedures which use class labels to find the directions in the parameter space that are most suited to separate the different classes. The simplest of such methods employs Wilks' lambda (Surendiran and Vadivel, 2011) statistic, which is computed as follows:

$$\lambda_W = \frac{S_{\text{wc}}}{S_{\text{total}}}, \tag{1}$$

where S_{wc} is the sample variance within class and S_{total} is the total sample variance along a given dimension. The statistic is computed for each dimension independently and the dimensions where it is close to zero are kept. An obvious disadvantage of this method is that it evaluates all dimensions independently, therefore it will fail if the data are separable along the linear combinations of the dimensions. In this case it is beneficial to use PCA transformation (without throwing away any dimensions) on the data before computing λ_W.

2.1. Linear discriminant analysis. In turn, linear discriminant analysis (McLachlan, 2004) is one of the most widely used dimension reduction methods. Its basic assumption is that the classes are normally distributed with different means but the same covariance matrix. Similarly to PCA, LDA computes optimal orthogonal linear combinations of the original dimensions. However, these new base vectors maximize separability of the classes instead of the variance of the entire dataset.

LDA is formulated as an optimization problem:

$$\max_{\boldsymbol{w}} \frac{\boldsymbol{w}^T \mathbf{S}_{\text{b}} \boldsymbol{w}}{\boldsymbol{w}^T \mathbf{S}_{\text{w}} \boldsymbol{w}}, \tag{2}$$

$$\mathbf{S}_{\text{b}} = \sum_{i=1}^{C} (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T, \tag{3}$$

$$\mathbf{S}_{\text{w}} = \sum_{i=1}^{C} \sum_{j=1}^{n_i} (\boldsymbol{\mu}_i - \boldsymbol{x}_{i,j})(\boldsymbol{\mu}_i - \boldsymbol{x}_{i,j})^T, \tag{4}$$

where $\mathbf{S}_{\text{b}}$ is the between-class scatter matrix, $\mathbf{S}_{\text{w}}$ is the within-class scatter matrix, $\boldsymbol{\mu}$ is the mean of the data set, $\boldsymbol{\mu}_i$ is the mean of the i-th class, $\boldsymbol{x}_{i,j}$ is the j-th vector of the i-th class, n_i is the number of vectors in the i-th class and C is the number of classes. It is worth noting that $\mathbf{S}_{\text{w}}$ may be replaced with the total scatter matrix $\mathbf{S}_{\text{t}}$ because of the following equation:

$$\mathbf{S}_{\text{t}} = \mathbf{S}_{\text{b}} + \mathbf{S}_{\text{w}}, \tag{5}$$

where

$$\mathbf{S}_{\text{t}} = \sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T, \tag{6}$$

with n being the total number of vectors. This optimization criterion leads to a generalized eigenvalue-eigenvector problem, which in the case of an invertible $\mathbf{S}_{\text{w}}$ or $\mathbf{S}_{\text{t}}$ can be solved by performing eigendecomposition on $\mathbf{S}_{\text{t}}^{-1}\mathbf{S}_{\text{b}}$ or $\mathbf{S}_{\text{w}}^{-1}\mathbf{S}_{\text{b}}$ and taking the eigenvector belonging to the largest eigenvalue. More discriminant dimensions may be extracted by taking the eigenvectors corresponding to the second, third, etc. largest eigenvalues. Note that this solves a different, but very similar optimization problem (Cunningham and Ghahramani, 2015).

Numerous variations of LDA have been proposed, such as penalized discriminant analysis (PDA) (Hastie *et al*., 1995), which is a weighted version of LDA. Weights allow the algorithm to penalize unstable features, thus improving the robustness of the method. Another variant is nonparametric DA (NDA) (Fukunaga, 1990), which uses a nearest-neighbors approach to define the between-class scatter matrix to relax the normal assumption of LDA. Górecki and Łuczak (2013) used the Moore–Penrose pseudoinverse to generalize LDA to problems with few observations.

Another important problem with LDA is that it does not take the local geometry of the dataset into account (Chai *et al*., 2007). To overcome this issue, locality sensitive discriminant analysis (LSDA) (Chai *et al*., 2007) has been proposed, which discovers the local manifold structure of the dataset, and uses this information to find the optimal discriminating projection. Methods such as structured semi-supervised discriminant analysis (SSDA) (Yang and Yuan, 2009) employ similar strategies, while using the local manifold information to make use of unlabeled data as well.

Lastly, some researchers have used the kernel trick to extend LDA into nonlinear cases, such as the generalized discriminant analysis method (Baudat and Anouar, 2000). In the work of Harandi *et al*. (2011) a Grassmannian graph embedding framework is employed to implement kernel-based DA. However, none of these methods are applicable to the case of SC classes, since they all assume at least partly labeled data.

2.2. Subclass and mixture methods. In this section we discuss particular variations of LDA which make the assumption that the data of each class are generated by the several different normal distributions, or in other words, they assume a mixture of Gaussians (MoG) model. Our reason for discussing these variants separately is that they are relatively easy to extend to the problem of SC classes, since the only important difference is that in the latter instances from different subclasses might be present *at the same time*.

Some of the mixture methods (Hastie and Tibshirani, 1996; Yang and Ahuja, 2001) use the expectation maximization (EM) (Dempster *et al*., 1977) algorithm to estimate the underlying distributions of the classes, and then employ classic LDA to find the optimal discriminating projection. A drawback of these methods is that they are not applicable if the number of data points is too low (Zhu and Martinez, 2006).

A different approach, subclass discriminant analysis (SDA) (Zhu and Martinez, 2006), uses clustering to estimate the means of the underlying normal distributions of the subclasses. With the help of subclass means they define the between-subclass scatter matrix as follows:

$$\mathbf{S}_{\text{bs}} = \sum_{i=1}^{C}\sum_{j=1}^{H}(\boldsymbol{\mu}_{i,j} - \boldsymbol{\mu})(\boldsymbol{\mu}_{i,j} - \boldsymbol{\mu})^T, \tag{7}$$

where C is the number of classes, H is the number of subclasses, and $\boldsymbol{\mu}_{i,j}$ is the mean of the j-th subclass of the i-th class. SDA replaces the between-class scatter $\mathbf{S}_{\text{b}}$ with the between-subclass scatter $\mathbf{S}_{\text{bs}}$. The procedure also determines the number of clusters using a brute-force iteration from one to a user-defined maximum and selecting the number that maximizes classification accuracy.

A modified version of SDA called mixture subclass discriminant analysis (MSDA) (Gkalelis *et al*., 2011) computes the scatter matrix only between the subclasses of *different classes* in order to prevent the algorithm from preferring directions that can separate subclasses of the same class. The between-subclass scatter matrix is computed as follows:

$$\mathbf{S}_{\text{bsb}} = \sum_{i=1}^{C-1}\sum_{j=1}^{H_i}\sum_{k=i+1}^{C}\sum_{l=1}^{H_j}(\boldsymbol{\mu}_{i,j} - \boldsymbol{\mu}_{k,l})(\boldsymbol{\mu}_{i,j} - \boldsymbol{\mu}_{k,l})^T, \tag{8}$$

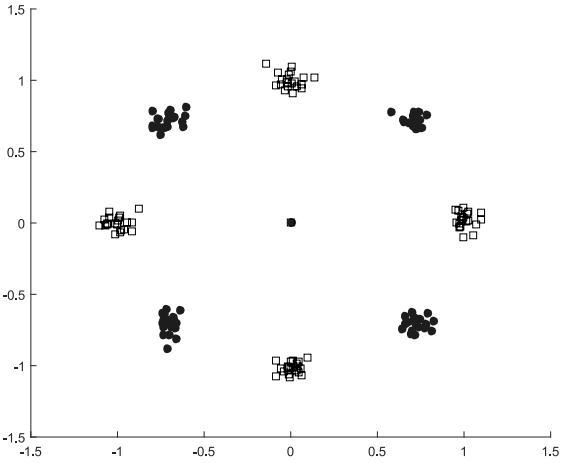
where H_i and H_j are the numbers of subclasses in the i-th and j-th classes. It is important to point out that SDA assumes that all classes have the same number of subclasses, since it uses the same H for all classes during clustering. This is assumption is false in many cases, and it may cause the algorithm to fail to separate subclasses if their number is underestimated for a given class.

3. Structured composite discriminant analysis

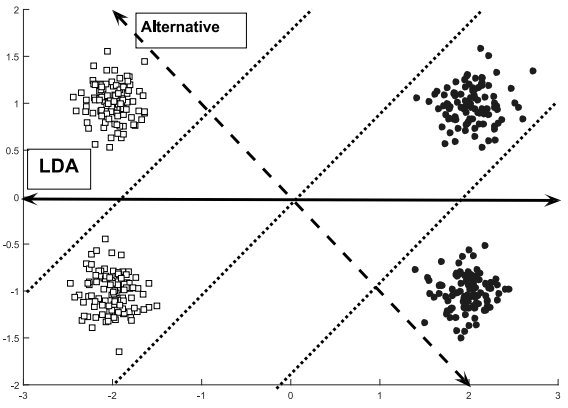
In this section we present our first method for performing discriminant analysis for SC classes. As mentioned before, SC classes represent a single instance of a class as a set of vectors, where an individual vector is called the node of an object. We assume that the nodes are drawn from normal distributions; however, all of the nodes within a single object are drawn from a different distribution. Also, *all* nodes in all objects of every class are in the same vector space. Lastly, we make no assumptions regarding the number of nodes per object; they may vary both between and within classes.

A central premise of the SCC problem is that although the nodes within a single object are different, there are similarities between certain nodes of distinct objects from the same class. However, there is no prior labeling available for these types of similarities. Nodes are only labeled according to which class and instance they belong to. In some cases spatial information might also be available for the nodes, which may be used by the application to estimate the pose of an object.

It is easy to see that it would be ill-advised to replace the nodes representing an instance of a class with their mean, since the difference between classes might be significant between individual nodes, while the classes have the same mean. It is also obvious that the distribution of all nodes in a class is not even approximately normal, therefore performing LDA on the level of nodes would produce suboptimal results as well. Lastly, classic LDA will not discriminate between the nodes of the same instance, which is an important requirement. These points are illustrated in Fig.1.



(a)



(b)

Fig. 1. Typical cases: separable nodes but indistinguishable means (a), where LDA fails to separate between nodes (b). The solid line is the dimension selected by LDA, while the dashed line shows a dimension that separates all subclasses.

3.1. Within-instance scatter matrix. The last problem of LDA we presented above can be addressed in a relatively straightforward way: by adding a second discriminant criterion that encourages selecting dimensions that separate nodes within instances. We call this addition the within-instance scatter matrix, which is computed as follows:

$$\mathbf{S}_{\text{wi}} = \sum_{i=1}^{C} \sum_{j=1}^{N_i} \sum_{k=1}^{n_{i,j}} (\boldsymbol{\mu}_{i,j} - \boldsymbol{x}_{i,j,k})(\boldsymbol{\mu}_{i,j} - \boldsymbol{x}_{i,j,k})^T, \quad (9)$$

where C is the number of classes, N_i is the number of instances in the i-th class, $n_{i,j}$ is the number of nodes in the j-th instance, $x_{i,j,k}$ is the k-th node of the j-th instance of the i-th class, and $\boldsymbol{\mu}_{i,j}^{\text{inst}}$ is the mean of nodes in the j-th instance of the i-th class. We can then define the between-class node scatter matrix similarly to classic LDA:

$$\mathbf{S}_{\text{bcn}} = \sum_{i=1}^{C} (\boldsymbol{\mu} - \boldsymbol{\mu}_i)(\boldsymbol{\mu} - \boldsymbol{\mu}_i)^T, \quad (10)$$

where $\boldsymbol{\mu}$ is the mean of all nodes and $\boldsymbol{\mu}_i$ is the mean of all nodes in the i-th class. Then the optimization criterion of structured composite discriminant analysis (SCDA) can be written as

$$\max_{\boldsymbol{w}} \frac{\boldsymbol{w}^T \mathbf{S}_{\text{bci}} \boldsymbol{w}}{\boldsymbol{w}^T \mathbf{S}_{\text{t}} \boldsymbol{w}}, \quad (11)$$

$$\mathbf{S}_{\text{bci}} = \mathbf{S}_{\text{bcn}} + \mathbf{S}_{\text{wi}}. \quad (12)$$

It is important to note that in some cases the two scatter matrices might not be in the same order of magnitude, which may lead to one of the criteria being suppressed in the favor of the other. In such cases, it is desirable to weigh the two scatter matrices to be added. The relative weight might be determined manually or by iterating through possible values. A special value that ensures the equal importance of the two criteria is

$$\mathbf{S}_{\text{bci}} = \mathbf{S}_{\text{bcn}} + \frac{\text{tr}(\mathbf{S}_{\text{bcn}})}{\text{tr}(\mathbf{S}_{\text{wi}})} \mathbf{S}_{\text{wi}}. \quad (13)$$

3.2. Rank adjustment. In Section 2 we explained that if the total scatter matrix of a dataset is invertible (which it usually is), then the generalized eigenvalue-eigenvector problem is reduced to a standard eigendecomposition performed on the discriminatory matrix $\mathbf{D}_{\text{b}} = \mathbf{S}_{\text{t}}^{-1}\mathbf{S}_{\text{b}}$, and selecting the eigenvectors corresponding to the few largest eigenvalues. With LDA and other classical methods, determining the exact number of dimensions to use is relatively simple. Since we know that,

$$\text{rank}(\boldsymbol{AB}) \leq \min(\text{rank}(\boldsymbol{A}), \text{rank}(\boldsymbol{B})), \quad (14)$$

and that the equality holds if one of the matrices is nonsingular. Since $\mathbf{S}_{\text{t}}^{-1}$ usually has a maximal rank, we can conclude that the discriminatory matrix has a rank that is equal to the rank of $\mathbf{S}_{\text{b}}$. According to (3) the between-class scatter matrix is computed as a sum of C dyads, which all have a rank of one. We also know that

$$\text{rank}(\boldsymbol{A} + \boldsymbol{B}) \leq \text{rank}(\boldsymbol{A}) + \text{rank}(\boldsymbol{B}), \quad (15)$$

therefore the rank of $\mathbf{S}_\mathrm{b}$ is less or equal to the number of dyads. However, in the case of discriminant analysis the between-class scatter matrix looses another degree of freedom, since $\boldsymbol{\mu}$ is usually computed from the dataset, which means that the dyads are not independent. Note that the same logic applies to the subclass and mixture subclass methods, except that the ranks are determined by the number of subclasses in those cases.

The rank of the discriminatory matrix is relevant because it determines the number of nonzero eigenvalues and sets an upper cap on the number of dimensions to select. Since the eigenvalues of $\mathbf{D}_\mathrm{b}$ can be understood as the magnitude of (linear) discriminatory information contained by each dimension, by selecting all eigenvectors that correspond to nonzero eigenvalues, we can compress all discriminatory information into a low dimensional feature space.

In the case of SCDA the within-instance scatter matrix is also a sum of dyads; the number of dyads, however, is relatively large. In practice it is safe to assume that the number of dyads is considerably higher than that of the original dimensions, which likely results in a nonsingular matrix. This causes several problems: First, determining the number of dimensions to keep is no longer trivial, since all eigenvalues are likely nonzero (Fig. 2). As a result, the method might select too many dimensions, which leads to computational inefficiency. Second, several nonzero eigenvalues of the within-instance scatter matrix are due to random variations in the data. When adding it to the between-class scatter matrix, these values might dominate the actual information contained in $\mathbf{S}_\mathrm{b}$. If the relative weight of the within-instance scatter is set so low that eigenvalues resulting from noise are negligible, then the discriminatory information represented in $\mathbf{S}_\mathrm{wi}$ is also weighed by a very small value. In other words, the relative weighting of the two matrices is problematic because the within-instance scatter has a considerably better condition number.

Luckily, this problem is manageable by compressing the within-class discriminatory information. In order to do this, we introduce the between-class and within-instance discriminatory matrices, $\mathbf{D}_\mathrm{b} = \mathbf{S}_\mathrm{t}^{-1}\mathbf{S}_\mathrm{b}$ and $\mathbf{D}_\mathrm{wi} = \mathbf{S}_\mathrm{t}^{-1}\mathbf{S}_\mathrm{wi}$, respectively. Using these, SCDA can be rewritten as

$$\mathbf{S}_\mathrm{t}^{-1}(\mathbf{S}_\mathrm{b} + \mathbf{S}_\mathrm{wi}) = \mathbf{D}_\mathrm{b} + \mathbf{D}_\mathrm{wi}. \tag{16}$$

By performing eigendecomposition on $\mathbf{D}_\mathrm{wi}$ and setting its smaller eigenvalues to zero we produce a new discriminatory matrix that only contains the most relevant separating information. The basic principle of this method is similar to PCA, except that here we find an optimal compression of the discriminatory information, as opposed to *all* information in the dataset.

3.3. Selecting the number of dimensions. The method discussed in the previous section leaves one last gap to fill in, and that is determining the number of eigenvalues to keep in the within-instance discriminatory matrix. A commonly applied practical heuristic is to look for a breakpoint in the graph of eigenvalues either visually or automatically. A similar approach is to set the ratio between the sum of eigenvalues retained and the sum of all eigenvalues, effectively setting an upper cap on the discriminatory information lost in the operation.

These methods, however, require arbitrary decisions on the part of the researcher. Therefore, we provide two simple methods that can be applied automatically. In the first one, we make use of the mathematical properties discussed in the previous section, namely, that you can discriminate between n classes using $n-1$ directions. The same is true if we wish to discriminate between n subclasses within the same class. If we can estimate the number of nodes per object H_i for all C classes (which is difficult if the number of nodes varies *within* the classes), then the number of eigenvalues to retain is the following:

$$N_\mathrm{r} = \sum_{i=1}^{C} H_i - C. \tag{17}$$

With this method, however, we run the risk of selecting too many dimensions, since this only provides an upper cap to the dimensions actually needed. Therefore, in our second method we evaluate the subsequent classification algorithm on our dataset for all possible values of N_r and select the value that provides the best result. While this method guarantees finding the best possible value, it does so at the cost of greatly increased computational requirements.

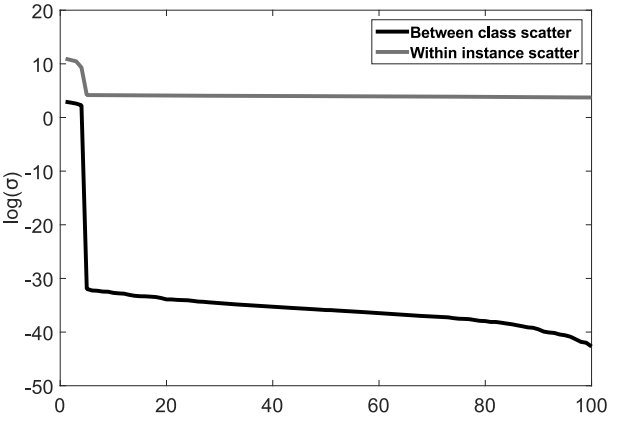


Fig. 2. Singular values of the between-class and within-instance scatter matrices. With the between-class scatter matrix, a clear breakpoint is visible in the singular values, while the break is much less pronounced in the within-instance scatter matrix.

4. Subclass SCDA

4.1. Improved clustering for SDA. Despite the addition of the within-instance scatter, SCDA still suffers from the inaccuracy of the between-class scatter matrix due to the invalid normal assumption. This problem has been, however, solved by mixture models and SDA by assuming a mixture of Gaussian models and computing the between-subclass scatter accordingly. It is important to recognize that SDA is fairly easy to apply to the problem of structured composite classes, especially the variants that discriminate between the subclasses of the same class as well.

In spite of the apparent simplicity of the solution, there are several issues with this approach. First, in certain cases the clustering step of SDA might not put nodes of a given instance into different subclasses. This might occur if there are similar nodes within instances but large differences between different instances. Secondly, SDA might not guess the number of nodes (subclasses) correctly, since it selects the number that optimizes separability of classes, not the separability of nodes within instances. Moreover, since SDA assumes that all classes have the same number of nodes, it will certainly fail to find the optimal separation in cases where classes have a different number of nodes.

Lastly, SDA uses a brute force approach to determine the correct number of subclasses, which means performing several clustering and discriminant analysis steps on high dimensional data. This is computationally expensive and should be avoided if possible.

Fortunately, it is possible to address all of these issues by improving the clustering procedure (SDA-IC). It is possible to estimate the number of subclasses in each class separately by setting them to the number of nodes in the largest instance of the given class. We may initialize subclass means by setting them to the nodes of the largest instance. Then, by using a nearest mean approach we can assign the remaining nodes to the different subclasses. This is equivalent to using a single iteration of the k-means clustering algorithm. During the assignment step it is possible to penalize the algorithm for putting two nodes of the same object into the same subclass cluster.

Since we initialized the clusters by considering our requirement of separating the nodes of a single instance, the resulting clusters are much more likely to be close to the optimum. Using this trick we used the additional information in our dataset to achieve significantly more accurate subclass clusters at reduced computational cost.

4.2. Combining SDA-IC and SCDA. A minor issue with the SDA-IC algorithm is that it compresses the two separability criteria into the same scatter matrix. This prevents us from scaling the relative importance of the two matrices which may be important for the reasons given in Section 3.1. An additional reason for doing this is that certain applications demand significantly less tolerance for one type of error than the other. This may justify adding artificial bias to the DA algorithm.

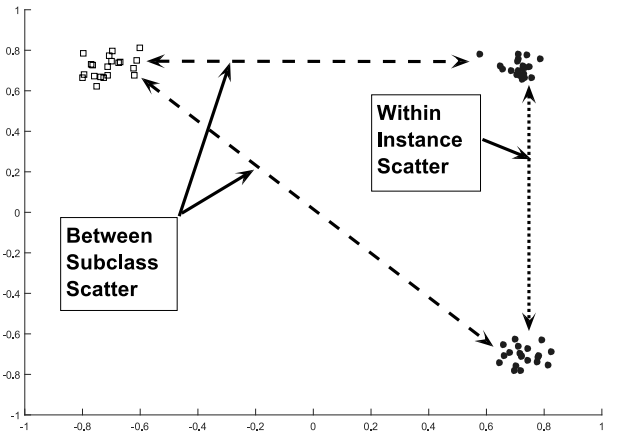


Fig. 3. Simple case where the vertical dimension, which only required to separate between nodes of the same class, is included in the between-subclass scatter.

This may be achieved by combining the SDA-IC and SCDA methods by defining the $\mathbf{S}_{\text{bci}}$ matrix of SSCDA as:

$$\mathbf{S}_{\text{bci}} = \mathbf{S}_{\text{bcb}} + \mathbf{S}_{\text{wi}}, \tag{18}$$

where $\mathbf{S}_{\text{bsb}}$ is the between-subclass scatter given by (8) and the actual subclasses are determined by SDA-IC.

It is important to recognize, however, that values resulting from the within-instance scatter may remain in the between-subclass scatter matrix, even if the MSDA definition is used. This is because the scatter is computed between a given subclass and *all* the subclasses from the other class. The phenomenon is illustrated in Fig. 3.

Our argument is that with the introduction of the within-instance scatter matrix it is sufficient to compute the between-subclass scatter only between the *closest* subclasses of different classes (in case of ambiguity all 'close' subclasses will be used). The reason for this is that the within-instance scatter matrix will include all the other scatter information that is required to separate subclasses. Therefore, this modification will not harm the algorithms' ability to discriminate.

It brings two advantages, however. First, using this new definition we could completely separate the two criteria into two scatter matrices, which can be weighted according to the applications' requirements. Second, during the classification of SC classes their nodes have to be classified first. During the classification of the individual nodes only similar nodes from the other classes may be responsible for misclassification. Therefore, discriminating between these nodes is far more important than discriminating between dissimilar ones. This way,

the between-subclass scatter matrix only contains the most important information, while the within-instance scatter contains secondary information.

It is worth noting that without the within-instance scatter matrix the SSCDA method would likely select a projection which ignores the dimensions that separate given nodes from relatively distant ones in other classes, resulting in a deeply flawed solution. The steps to perform SSCDA are given in Algorithm 1.

Algorithm 1. Steps of SSCDA.

1. Determine clusters using instance labels.
2. Compute $\mathbf{S}_{\mathrm{t}}$.
3. Compute $\mathbf{S}_{\mathrm{wi}}$ using instance labels.
4. Compute $\mathbf{S}_{\mathrm{bsb}}$ from clusters, using only the nearest subclasses.
5. $\mathbf{D}_{\mathrm{b}} = \mathbf{S}_{\mathrm{t}}^{-1}\mathbf{S}_{\mathrm{bsb}}$ and $\mathbf{D}_{\mathrm{wi}} = \mathbf{S}_{\mathrm{t}}^{-1}\mathbf{S}_{\mathrm{wi}}$.
6. Adjust the rank of $\mathbf{D}_{\mathrm{wi}}$ using SVD.
7. Perform LDA using the matrix $\mathbf{D}_{\mathrm{b}} + \mathbf{D}_{\mathrm{wi}}$.

4.3. Extension of Wilks' lambda. In the last part of the current section we show that it is possible to use the ideas developed in the previous sections to provide an extension of Wilks' lambda statistic. This is done by creating two different lambdas: one for between-class and the other for within-instance separability. Since

$$S_{\mathrm{wc}} = S_{\mathrm{bi}} + S_{\mathrm{wi}}, \tag{19}$$

where S_{wc} is the within-class scatter, S_{bi} is the scatter of instance means, and S_{wi} is a within-instance scatter, we can use a statistic for within-instance separability that is computed as follows:

$$\lambda_{\mathrm{wi}} = \frac{S_{\mathrm{bi}}}{S_{\mathrm{wc}}}. \tag{20}$$

The lambda statistic used for between-class separation can be computed simply using the classic formulation (1). Here, the within-group scatter S_{wg} contains within-subclass scatter values. In order to select dimensions using these two criteria one might use the sum or the weighted sum of the two lambda values $\lambda_{\mathrm{SSCDA}} = \lambda_{\mathrm{bsc}} + \mu\lambda_{\mathrm{wi}}$, or simply combine the dimensions selected by using the two lambda statistics separately.

5. Experimental results

In this section we present the evaluation procedure we used to assess the efficiency of our methods. We have compared the performance of LDA, SDA, SDA-IC, SCDA, SSCDA, and the Wilks lambda method on several different datasets. The methods used in evaluation are summarized in Table 1. We used 10-fold cross-validation for computing validation accuracies. We begin by presenting our classification method, followed by results on different types of datasets.

Table 1. Methods used for comparison.

Method	Description
LDA	As given in Section 2.1
SDA	Uses the between-class scatter matrix
SDA-IC	SDA with the improved clustering method
SCDA	LDA with the addition of the within-instance scatter matrix
SSCDA	Combination of the two previous methods
Wilks	Uses the Wilks lambda statistic for feature selection

Our classification method is a semi-supervised one, since we assume that the nodes of instances are not labeled. This means that we have to construct a class model first that can be used to classify nodes individually. After running the DA algorithm we use the k-medoids clustering method to create node clusters, since it is more robust in the case of outlier data points. Unlike during discriminant analysis, this clustering step determines the number of clusters automatically by minimizing the within-group scatter, while penalizing the number of clusters.

In the next step, nodes are labeled according to their distance to the model clusters. Each node is given two different labels: the first is the class label, while the second is the label of the node cluster within the given class. The class label of each node may be interpreted as a vote for the presence of a certain class. The class label of the object is then decided on by the plurality of node votes. The node cluster labels can be used to discriminate between nodes of a single instance.

We compute two important statistics to measure the methods' performance on the two criteria. The first, a_{wi}, is the percentage of nodes that have a different node label within the correct class. This measures the algorithm's ability to discriminate between nodes in the same object instance. The second, a_{c}, is the percentage of object instances that were assigned to the correct class.

5.1. Random synthetic classes. Since there is not a great number of online databases that contain SC classes, we first tested the efficiency of our method on synthetic datasets. The first dataset contains five random classes, each with five different random nodes that have unique probabilities to appear in a given instance of the class. There are four variations of this dataset: In two cases there is no overlap between dimensions that are useful for separating classes and those that can be used to discriminate between nodes of a particular instance. In the remaining two cases the same dimensions can be used for both kinds of separation. Also, in all four datasets we add noise to the nodes, drawn from a normal distribution with

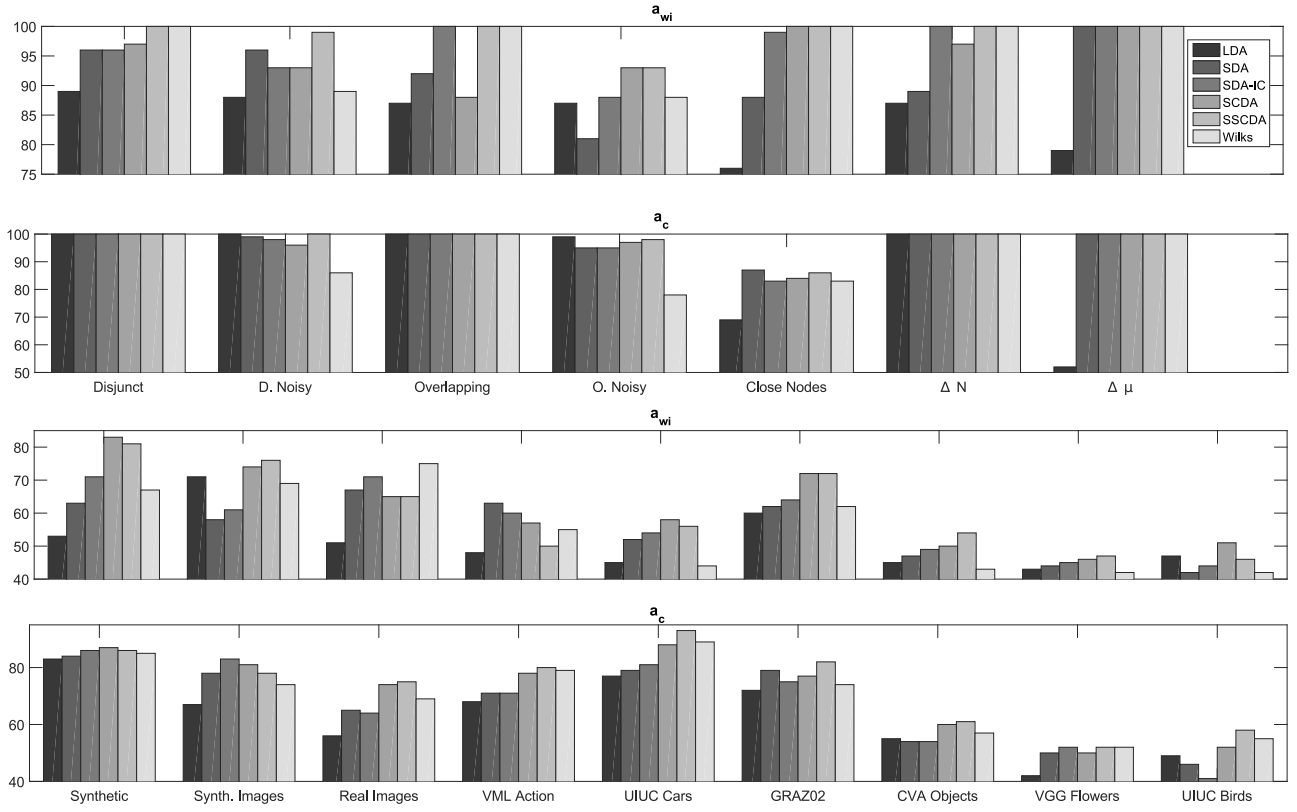


Fig. 4. Results of our method for synthetic datasets (top) and for shape-graph and image datasets (bottom).

zero mean, but in two cases the noise is relatively small, allowing easy separation, while in the other two cases the variance of the noise is comparable to that between the nodes and classes. The datasets are: disjunct, disjunct noisy, overlapping, and overlapping noisy.

The results (Fig. 4) demonstrate that our methods outperform the standard ones for random structured composite classes. The one exception is the overlapping noisy dataset, where our methods trade some of the classification accuracy for better within-instance separation.

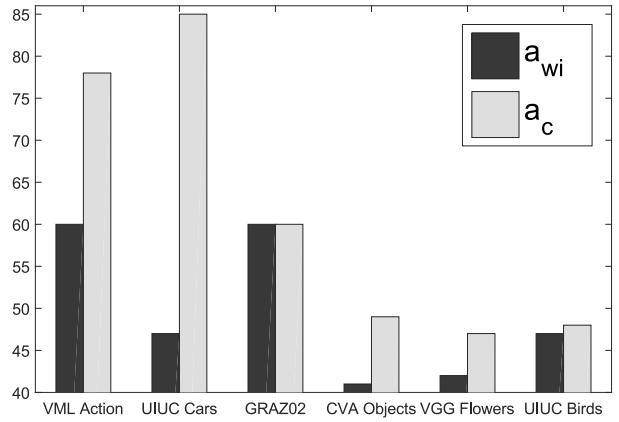


Fig. 5. Performance of SURF features on image datasets.

In addition to the previous four datasets we created examples for peculiar situations in which the classic methods clearly fail. One such case is when the closest subclasses from different classes are very close compared with other subclasses (close nodes dataset). If other subclasses contain significant noise in the directions that can separate the closest subclasses then all methods with the exception of SSCDA will perform poorly. Another case where LDA and SDA give suboptimal results is when there are classes with a different number of nodes (ΔN). We already mentioned a third case in Section 4, where the classes have the same mean but different subclass means, causing LDA to fail ($\Delta\mu$ dataset). We created datasets to illustrate all three cases, each containing two different classes with 500 instances of both classes.

The results (Fig. 4) for the special dataset mostly meet our expectations. The CN and ΔN datasets are shown to confuse LDA and SDA, while our methods perform remarkably well on them. The $\Delta\mu$ dataset proves to be difficult for LDA, exactly as predicted in Section 3.

5.2. 3D shape-graphs. In the second stage of testing we use 3D shape recognition as an illustration of our method's efficiency. We created three different databases

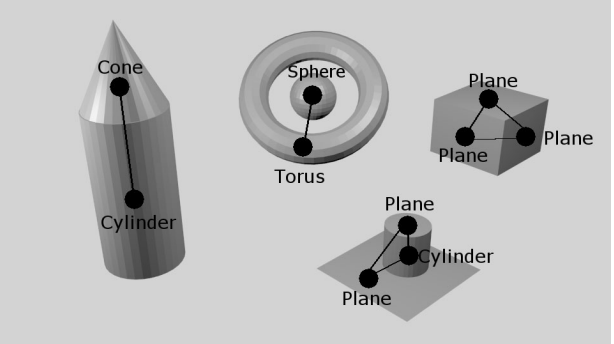


Fig. 6. Graph constructed from primitive shapes (only significant edges are drawn).

Table 2. Primitives, their features and reference frames.

Primitive	Features	Reference frame
Plane	Area Diameter Bounding box area	Centroid Normal
Sphere	Radius	Centroid
Cone	Radius Height Angle	Peak Axis
Cylinder	Radius Height	Centroid Axis
Torus	Inner radius Body radius	Center Axis

for testing. The first one uses synthetic shape descriptors with random noise added, while the second one is made from synthetic images created in Blender. The third dataset employs real image series of simple objects. In both image-based cases we introduced significant variations within object classes. The two synthetic databases include around 2000 shapes in five classes, while a real-world database includes roughly 8000 shapes in five classes.

For our test we use a shape description method that approximates the 3D shape of an object by a graph primitive shapes, where each node of the graph represents a shape while the edges describe the geometric relationship between them. The advantage of using this approach is that it is able to represent both global and local geometry well (Szemenyei and Vajda, 2015).

In both image-based cases 3D reconstruction was performed using a multi-view technique (Wu, 2013) with 15 successive images of the objects. Then, the 3D point cloud is segmented into primitive shapes, for which we use the efficient RANSAC algorithm developed by Schnabel *et al*. (2007). The graph constructed from the 3D scene (Fig. 6) contains the following five primitive shapes: spheres, cones, cylinders, planes and tori.

In the next part of the description algorithm, we assign features to the nodes and edges of the graph. For the nodes of the graph (which represent primitives) we compute features, using the point cloud of inliers determined by the RANSAC algorithm. Since there are five primitive shape categories with different possible features we would need to construct separate feature vectors for each primitive type, which would make comparing nodes difficult. In order to avoid this, we construct just one feature vector for nodes by concatenating the feature vectors for the different primitive types. The features that belong to another primitive type are set to zero.

It is important to note that it is possible to assign a coordinate system to all primitive shapes that may be used to describe geometric relations between given nodes. However, almost all shapes have symmetric properties, therefore these coordinate systems will be ambiguous. Nevertheless, we can still assign an origin to all shapes unambiguously and a direction to all primitives, with the exception of the sphere. The features and the reference frames are shown for each primitive type in Table 2.

The edges of the graph represent the relations of these coordinate systems. We store the translation and the rotation between the two coordinate systems. Because of the ambiguity of these coordinate systems, we determine the smallest possible rotation between the two vectors representing the directions. Also, since the world coordinate system might be different for different scenes, we use only the magnitude of translation and the angle of rotation between to coordinate systems as features.

Another essential property of our shape description algorithm is that we always create full graphs. This way we need no arbitrary threshold to determine the adjacency of nodes, since this property has already been encoded in the edge features. Moreover, this way the adjacency property remains a real value instead of a binary one, which allows a more robust solution.

The last step of our shape description algorithm is to convert the graph of vectors representation into a set of vectors by embedding the nodes of the graph into a vector space. Since in this case the nodes and edges of the graph are vectors instead of real weights, methods such as the one developed by Demirci *et al*. (2011) are not applicable. Therefore, we construct a descriptor vector by ordering the nodes of the graph by the distance from the node we want to embed and concatenating their descriptor vectors. In order to limit the size of the descriptor vector we only consider the N closest nodes. We can also add edge features to the descriptor vector by concatenating them to the descriptor vector in the same order.

The results (Fig. 4) indicate that our methods are able to provide better results on 3D shape graphs. The difference between the algorithms is much more subtle on the synthetic dataset, which is likely due to lower noise.

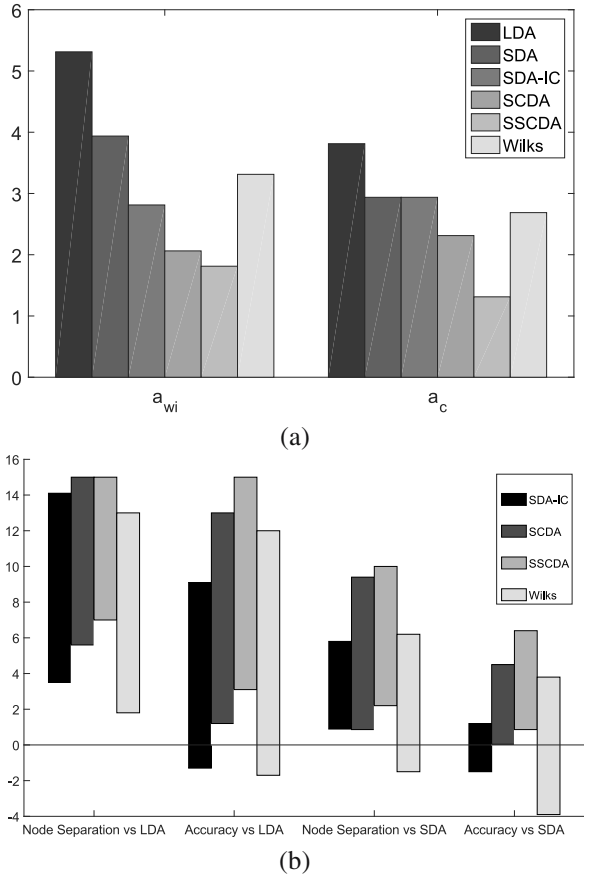


Fig. 7. Average ranks of the different methods (a), 95% credible interval of the effect of our methods (b).

5.3. Feature-based object recognition. The third problem we applied our discriminant analysis algorithms to was visual object classification. Here, we constructed structured composite classes by detecting 15 strongest SURF features on each image and storing them as nodes of an instance of the given class. However, instead of using the SURF descriptors we vectorized the 32×32 image patch surrounding the key-points. Thus our nodes consist of 1024 raw pixel values. We performed this on several object recognition databases, including datasets with few classes, such as the VML Action Class Database (Wang *et al.*, 2006), the UIUC Image Database for Car Detection (Agarwal *et al.*, 2004), the GRAZ dataset (Marszałek and Schmid, 2007), and sets with numerous classes, such as the Caltech Computational Vision Archive (CVA) (Fei-Fei *et al.*, 2007) and the VGG Flower Database (Nilsback and Zisserman, 2008), and the UIUC Bird database (Lazebnik *et al.*, 2005).

The results (Fig. 4) demonstrate the efficiency of our methods for image classification, with SCDA and SSCDA consistently outperforming the other methods. Interestingly, SDA, SDA-IC and the Wilks method are bested by LDA on some datasets. We also evaluated the performance of our classification method by using full SURF features as a baseline (Fig. 5). When using SURF features, no dimension reduction was performed.

5.4. Statistical analysis. In this section we perform statistical analysis of the results in order to illustrate the efficiency of our methods. The first measure used is the average ranks of the methods on all datasets, computed separately for the values a_{c} and a_{wi}.

In the next part of our analysis we compute the 95% credible intervals (CIs) for the size of the improvement brought by our methods. To get credible intervals we use a Bayesian alternative of the paired samples t-test (Bååth, 2014) based on BEST (Kruschke, 2013). We compute this statistic for both a_{wi} and a_{c} while comparing our methods against LDA and SDA. The results (Fig. 7) show that while our methods tend to perform better or the same as LDA or SDA, only SCDA and SSCDA have a *credibly* positive effect on both measures against LDA and SDA. Also, we can compare the two best methods, getting the credible intervals $[-1.4, 3.2]$ for a_{wi} with a 76.9% probability that SSCDA outperforms SCDA on node separation, and $[0.18, 2.8]$ for a_{c} with a 98.7% probability that SSCDA performs better in classification accuracy. This allows us to conclude that SSCDA is the superior method of the two.

We also compared the SURF descriptor to SSCDA using the same Bayesian test. The results are $[-6, 16]$ for a_{wi} with a 85.6% probability that SSCDA outperforms the SURF descriptors at node separation, and $[0.88, 18]$ for a_{c} with a 98.2% probability that SSCDA performs better at classification accuracy. This proves that SSCDA is a good way to extract features from an image, since it credibly outperforms SURF at classification accuracy. This is unsurprising, since the SURF descriptor was not optimized for structured composite image classification.

5.5. Comparing methods for rank selection. In this section we compare our methods for selecting the rank of the within-instance scatter matrix. We evaluated our methods for all datasets, and computed the accuracy $((a_{\mathrm{wi}}+a_{\mathrm{c}})/2)$ and the number of dimensions kept. We use SSCDA and the Bayesian t-test to compare the different methods. We employ the iterative method as a baseline to compare the other methods against, since it evaluates all possible values and therefore it is bound to find the the best possible accuracy at the lowest number of dimensions.

Figure 8 shows the results of the test for the two criteria. The results justify adjusting the rank of the within-instance scatter, since without it our method produces clearly suboptimal results both in terms of classification accuracy and dimension reduction. Among rank adjustment methods, iterative trial and error clearly

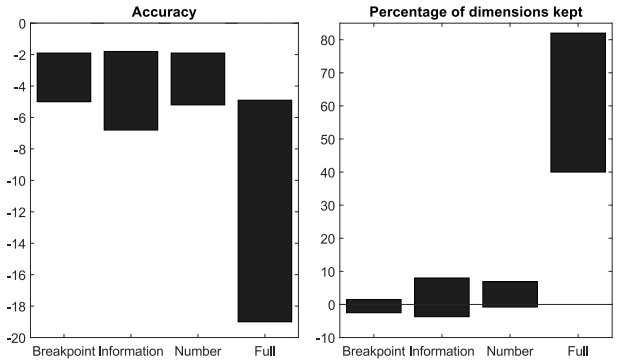


Fig. 8. 95% credible interval of the effect of rank selection methods.

outperforms all other methods on accuracy. It is worth noting that the other methods tend to keep a similar number of dimensions. This means that it may be a viable strategy to use one of these methods to find an initial value and check the surrounding few values for the optimum.

6. Conclusion

In this paper we presented methods for dimension reduction of structured composite (SC) classes by discriminant analysis. An extension of LDA called structured composite DA was introduced, followed by an extension of SDA called subclass structured composite DA. We also discussed improving the clustering method of SDA and extending the Wilks lambda statistic for SC classes.

We evaluated the efficiency of the methods using both synthetic and real-world datasets and compared them with the classic versions. We found that our additions significantly increased the efficiency of the original algorithms by taking the structure of the data into account. The results show that our methods are successful in achieving two criteria on both synthetic and real-world datasets. It was also demonstrated that SCDA and SSCDA usually outperform the usual methods on structured composite classes.

References

Agarwal, S., Awan, A. and Roth, D. (2004). Learning to detect objects in images via a sparse, part-based representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26**(11): 1475–1490.

Bååth, R. (2014). Bayesian first aid: A package that implements Bayesian alternatives to the classical *.test functions in R, *International R User Conference UseR! 2014, Los Angeles, CA, USA*, pp. 86.

Baudat, G. and Anouar, F. (2000). Generalized discriminant analysis, *Neural Computation* **12**(1): 2385–2404.

Boutsidis, C., Zouzias, A., Mahoney, M.W. and Drineas, P. (2011). Stochastic dimensionality reduction for k-means clustering, *CoRR* **abs/1110.2897**, `http://arxiv.org/abs/1110.2897`.

Bronstein, A.M., Bronstein, M.M. and Ovsjanikov, M. (2010). Feature-based methods in 3D shape analysis, *in* N. Pears *et al.* (Eds.), *3D Imaging Analysis and Applications*, Springer-Verlag, London, pp. 185–216.

Chai, D., He, X., Zhou, K., Han, J. and Bao, H. (2007). Locality sensitive discriminant analysis, *International Joint Conference on Artificial Intelligence, Hyderabad, India*, pp. 708–713.

Cunningham, J.P. and Ghahramani, Z. (2015). Linear dimensionality reduction: Survey, insights, and generalizations, *Journal of Machine Learning Research* **16**(1): 2859–2900.

Demirci, M.F., Osmanlioglu, Y., Shokoufandeh, A. and Dickinson, S. (2011). Efficient many-to-many feature matching under the l1 norm, *Computer Vision and Image Understanding* **115**(7): 967–983.

Dempster, A.P., Laird, N.M. and Rubin, D.B. (1977). Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society* **39**(1): 1–38.

Fei-Fei, L., Fergus, R. and Perona, P. (2007). Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories, *Journal Computer Vision and Image Understanding* **106**(1): 59–70.

Fei-Fei, L. and Perona, P. (2005). A Bayesian hierarchical model for learning natural scene categories, *IEEE Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA*, pp. 524–531.

Felzenszwalb, P.F., Girshick, R.B., McAllester, D. and Ramanan, D. (2010). Object detection with discriminatively trained part-based models, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32**(9): 1627–1645.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*, Academic Press, San Diego, CA.

Gkalelis, N., Mezaris, V. and Kompatsiaris, I. (2011). Mixture subclass discriminant analysis, *IEEE Signal Processing Letters* **18**(5): 319–322.

Górecki, T. and Łuczak, M. (2013). Linear discriminant analysis with a generalization of the Moore–Penrose pseudoinverse, *International Journal of Applied Mathematics and Computer Science* **23**(2): 463–471, DOI: 10.2478/amcs-2013-0035.

Harandi, M.T., Sanderson, C., Shirazi, S. and Lovell, B.C. (2011). Graph embedding discriminant analysis on Grassmannian manifolds for improved image set matching, *IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, CO, USA*, pp. 2705–2712.

Hastie, T., Buja, A. and Tibshirani, R. (1995). Penalized discriminant analysis, *Annals of Statistics* **23**(1): 73–102.

Hastie, T. and Tibshirani, R. (1996). Discriminant analysis by Gaussian mixtures, *Journal of the Royal Statistical Society* **58**(1): 155–176.

Hyvärinen, A., Karhunen, J. and Oja, E. (2001). *Independent Component Analysis*, John Wiley and Sons, New York, NY.

Jolliffe, I.-T. (2002). *Principal Component Analysis*, Springer-Verlag, New York, NY.

Kruschke, J.K. (2013). Bayesian estimation supersedes the t-test, *Journal of Experimental Psychology: General* **142**(2): 573–603.

Kulczycki, P. and Łukasik, S. (2014). An algorithm for reducing the dimension and size of a sample for data exploration procedures, *International Journal of Applied Mathematics and Computer Science* **24**(1): 133–149, DOI: 10.2478/amcs-2014-0011.

Kumar, C.A. (2009). Analysis of unsupervised dimensionality reduction techniques, *Computer Science and Information Systems* **6**(2): 217–227.

Lazebnik, S., Schmid, C. and Ponce, J. (2005). A maximum entropy framework for part-based texture and object recognition, *IEEE International Conference on Computer Vision, Beijing, China*, pp. 832–838.

Lin, T.-S. (1992). Statistical feature extraction and selection for IC test pattern analysis, *IEEE International Symposium on Circuits and Systems, San Diego, CA, USA*, pp. 391–394.

Liu, X., Wang, Z., Liu, J. and Feng, Z. (2008). Face recognition with locality sensitive discriminant analysis based on matrix representation, *IEEE International Joint Conference on Neural Networks, Montreal, Canada*, pp. 4052–4058.

Lowe, D.G. (2004). Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision* **60**(2): 91–110.

Marszałek, M. and Schmid, C. (2007). Accurate object localization with shape masks, *IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA*, pp. 1–8.

McLachlan, G.-J. (2004). *Discriminant Analysis and Statistical Pattern Recognition*, John Wiley and Sons, New York, NY.

Nilsback, M.-E. and Zisserman, A. (2008). Automated flower classification over a large number of classes, *Indian Conference on Computer Vision, Graphics and Image Processing, Bhubaneswar, India*, pp. 722–729.

Schnabel, R., Wahl, R. and Klein, R. (2007). Efficient RANSAC for point-cloud shape detection, *Computer Graphics Forum* **26**(2): 214–226.

Schnabel, R., Wessel, R., Wahl, R. and Klein, R. (2008). Shape recognition in 3D point-clouds, *16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Pilzen, Czech Republic*, pp. 65–72.

Song, F., Z.Guo and Mei, D. (2010). Feature selection using principal component analysis, *International Conference on System Science, Engineering Design and Manufacturing Informatization, Guiyang, China*, pp. 27–30.

Surendiran, B. and Vadivel, A. (2011). Feature selection using stepwise ANOVA discriminant analysis for mammogram mass classification, *ACEEE International Journal on Signal and Image Processing* **2**(1): 17–19.

Szemenyei, M. and Vajda, F. (2015). Learning 3D object recognition using graphs based on primitive shapes, *Workshop on the Advances of Information Technology, Budapest, Hungary*, pp. 187–195.

Wang, Y., Jiang, H., Drew, M.S., Li, Z.-N. and Mori, G. (2006). Unsupervised discovery of action classes, *IEEE Conference on Computer Vision and Pattern Recognition, New York, NY, USA*, pp. 1645–1661.

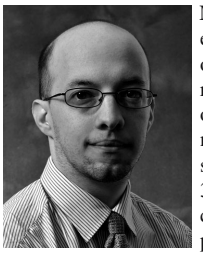
Wu, C. (2013). Towards linear-time incremental structure from motion, *International Conference on 3D Vision, Seattle, WA, USA*, pp. 127–134.

Yang, M.-H. and Ahuja, N. (2001). Face detection using multimodal density models, *International Series in Video Computing* **1**(1): 97–122.

Yang, M. and Yuan, X.-M. (2009). Structured semi-supervised discriminant analysis, *International Conference on Wavelet Analysis and Pattern Recognition, Baoding, China*, pp. 148–153.

Yasuoka, S., Kang, Y., Morooka, K. and Nagahashi, H. (2004). Texture classification using hierarchical discriminant analysis, *IEEE Conference on Systems, Man and Cybernetics, The Hague, The Netherlands*, pp. 6395–6400.

Zhu, M. and Martinez, A.M. (2006). Subclass discriminant analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**(8): 1274–1286.



Marton Szemenyei received an MSc degree in electrical engineering at the Budapest University of Technology and Economics in 2015. Currently, he is a PhD student at the Department of Control Engineering and Information Technology at the same university. His main research topics include tangible augmented reality, 3D shape recognition, and machine learning in computer vision. He has also published several papers on the application of information technology to environmental problems.

Ferenc Vajda received MSc degrees in electrical engineering (1998) and biomedical engineering (2001), as well as a PhD degree in electrical engineering (2006), all from the Budapest University of Technology and Economics. He is currently working as an associate professor at the Department of Control Engineering and Information Technology there. His main research activities focus on processing 2D/3D images, and virtual and augmented reality systems.

Received: 4 March 2016
Revised: 10 August 2016
Re-revised: 17 September 2016
Accepted: 6 October 2016