Linear dimension reduction for evolutionary data

E. Kokiopoulou[1], D. Kressner[1], Y. Saad[2]

Abstract

We consider the problem of linear dimensionality reduction for high-dimensional evolutionary data, whose distribution changes over time or with respect to a parameter. Supervised dimensionality reduction methods have proven to be very successful in real-world classification problems. When dealing with evolutionary data, we would like to perform classification that is not only robust to noise and short-term changes, but also adaptive to long-term drifts in the data distribution. In order to capture this notion of temporal smoothness, we propose the use of an additional penalty term in classic optimization-based formulations of dimensionality reduction. The penalty term prevents the reduced space in one time step to differ too much from the one in the previous time step. Experiments with synthetic and real-world data provide evidence that such an approach leads to improved classification performance.

Keywords: Dimensionality reduction, evolutionary data

1. Introduction

Dimensionality reduction of high-dimensional data plays a crucial role in a variety of research areas, including computer vision, data visualization, and pattern recognition. The goal of dimensionality reduction is to simplify the data and make it more amenable to further analysis. For this purpose, the high-dimensional data is mapped to a lower-dimensional space such that certain properties, such as locality and other geometric characteristics, are preserved. Principal component analysis (PCA) is among the most popular reduction techniques. It belongs to the class of linear methods, in the sense that the mapping from the original to the reduced data is linear. For the purpose of classifying data, however, PCA is not well suited due to its unsupervised nature. This observation has led to the development of a range of supervised linear methods for dimensionality reduction, which take into account the class labels of the training data. We mention, for example, Linear Discriminant Analysis (LDA) [1], Locality Preserving Projections (LPP) [2], Orthogonal Neighborhood Preserving Projections (ONPP) and Orthogonal Locality Preserving Projections (OLPP) [3]. The reader is referred to [4] for a recent overview of dimensionality reduction methods.

The focus of this paper will be on evolutionary data, whose distribution changes over time or with respect to a parameter. Using individual snapshots of the data to determine the classifier may not be optimal. If the classifier is based on a single snapshot, it may be too biased and sensitive to short-term noise, and therefore lead to poor performance. This is conceptually illustrated in Figures 1(a) and 1(b), where the classifier is adjusted to every snapshot and becomes too sensitive to the particular realization of the data set. Previous work by [5] and [6] has indicated that taking previous data snapshots into account can be rather helpful and be seen as a form of weak supervision. Figure 1(c) shows an example where the classifier obtained from both snapshots differs significantly from the classifiers obtained from the individual snapshots. To circumvent these problems with individual snapshots, one could be tempted to construct a classifier from the whole data trajectory instead. However, such a global approach may be infeasible in resource-constrained environments. Even if feasible, this approach suffers from the fact that the data can be expected to undergo significant changes over longer times, leading to meaningless global classifiers.

There is clearly a need for flexible learning algorithms that can adapt to the changes of the data distribution. Ideally, when dealing with evo-

Email address:
effrosyni.kokiopoulou@sam.math.ethz.ch, daniel.kressner@sam.math.ethz.ch, saad@cs.umn.edu (Y. Saad)

[1]Seminar for Applied Mathematics, ETH, HG G J49, Rämistrasse 101, 8092 Zürich, Switzerland.

[2]Department of Computer Science and Engineering; University of Minnesota; Minneapolis, MN 55455. Work supported by NSF under grants DMS-0810938 and DMR-0940218, and by the Minnesota Supercomputer Institute.

lutionary data, we would like to perform classification that is not only robust to noise and short-term changes, but also adaptive to long-term drifts in the data distribution. Since in most real world applications the concept drift to be learned rarely changes dramatically, classification should be done in a temporally smooth fashion; that is, the output of the classifier should not be too far from the output of the classifier corresponding to the recent history. Such temporally smooth approaches to data analysis have been used successfully in clustering [5] and semi-supervised learning [6].

To the best of our knowledge, there is no previous work on dimensionality reduction for evolutionary data. The aim of this work is to fill this gap. We propose a regularization framework for linear dimensionality reduction that imposes temporal smoothness on the reduced space to be learned at each time step. It is shown that the proposed framework admits algorithmic formulations based on eigenvalue problems. Our framework is illustrated for the dimensionality reduction methods OLPP and ONPP. However, it is worth emphasizing that the same regularization framework can be applied to other linear methods (such as PCA, LDA and so on). Finally, we would like to stress that dealing with evolutionary data is rather different from the scenario where *parts* of the training set are updated and one is interested in on-line or incremental algorithms that avoid re-training from scratch. In our scenario, the distribution of the *whole* data set changes at every time step.

The rest of this paper is organized as follows. Section 2 is a review of methods for linear dimensionality reduction. Section 3 is concerned with dimensionality reduction of evolutionary data, for which two different algorithmic formulations are proposed. Section 4 provides experimental results both on synthetic and on real-world data sets. Finally, some concluding remarks are given in Section 5.

2. Linear dimensionality reduction

In this section, we provide an overview of the two linear dimensionality reduction methods ONPP and OLPP that will be used as illustrative case studies in our evolutionary framework.

Consider a data set $X \in \mathbb{R}^{m\times n}$. Linear dimensionality reduction methods derive the low-dimensional samples $Y \in \mathbb{R}^{d\times n}$ (where $d \ll m$) from the high-dimensional samples X by means of a linear transformation

$$Y = V^\top X, \quad \text{where} \quad V \in \mathbb{R}^{m\times d}. \tag{1}$$

The matrix V is typically obtained as the solution of an optimization problem imposing that certain properties of the original data X are preserved in the reduced space. In particular, OLPP and ONPP build an *affinity graph* among the high-dimensional samples x_i and learn the matrix V such that the graph is preserved among the y_i. Each x_i becomes a node in the affinity graph and edges are drawn only between data samples that are close-by, usually in terms of the Euclidean distance. In the *supervised* versions of OLPP and ONPP, two data samples are considered adjacent if and only if they belong to the same class. Hence, in this case the affinity of samples is determined from class memberships and not from Euclidean distances.

It is common to assign weights to the edges of the affinity graph and the two methods OLPP and ONPP differ in the way these weights are defined.

2.1. OLPP (Orthogonal Neighborhood Preserving Projection))

A very popular choice of weights in OLPP consists of Gaussian weights

$$w_{ij} = e^{-\frac{\|x_i - x_j\|_2^2}{2\sigma^2}}, \tag{2}$$

where σ is a free parameter determining the range of influence between data samples. The affinity graph defines a sparse matrix $W \in \mathbb{R}^{n\times n}$, whose entries are the weights w_{ij} for adjacent nodes in the graph and zero otherwise.

OLPP aims at preserving the data affinity under reduction, that is, two close-by samples x_i and x_j should be mapped to y_i and y_j that are also close-by in the reduced space. For this purpose, the criterion

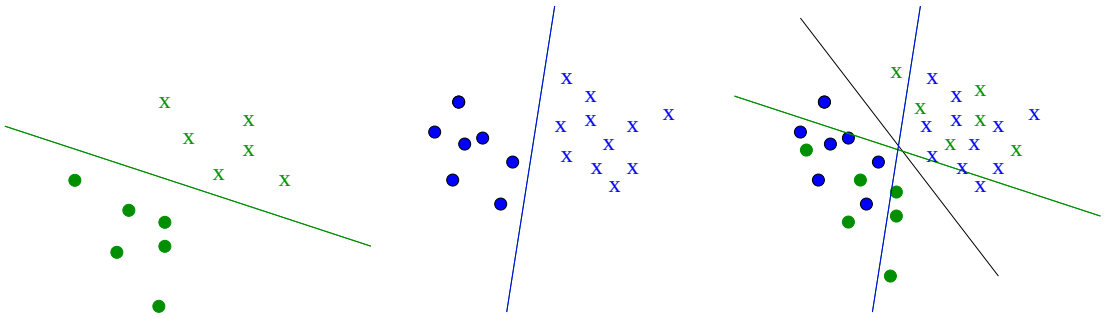
$$E_{\text{olpp}} = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\|y_i - y_j\|_2^2 \tag{3}$$

is minimized with respect to the low dimensional samples, enforcing y_i to lie close to y_j when the weight w_{ij} is high. Defining the Laplacian matrix $L := D - W$, where D is a diagonal matrix whose diagonal entries contain the row sums of W, and using (1), it has been shown in [3] that the criterion above can be rewritten as

$$E_{\text{olpp}} = \text{Tr}\,[YLY^\top] = \text{Tr}\,[V^\top XLX^\top V]. \tag{4}$$

In addition, OLPP imposes that the matrix V has orthonormal columns. In summary, the optimization problem solved by OLPP can be formulated as

$$\begin{cases} \min\limits_{V\in\mathbb{R}^{m\times d}} & \text{Tr}\left[V^\top XLX^\top V\right]. \\ V^\top V = I \end{cases} \tag{5}$$



(a) First snapshot t_1 (b) Second snapshot t_2 (c) Both snapshots

Figure 1: Illustration of robustness enhancement when combining data from several snapshots for constructing the classifier.

It turns out that the solution of (5) can be obtained from solving a standard eigenvalue problem. The solution V contains the eigenvectors belonging to the d smallest eigenvalues of the symmetric matrix $XLX^\top$, see [3] for more details.

2.2. ONPP (Orthogonal Neighborhood Preserving Projection)

ONPP constructs the weights w_{ij} of the affinity graph in a different way, such that each x_i can be locally reconstructed from the set of its nearest neighbors x_j. This is achieved by minimizing

$$E(W) = \sum_i \|x_i - \sum_j w_{ij} x_j\|_2^2$$

with respect to w_{ij}, under the constraint that W respects the sparsity pattern of the affinity graph. The weights can be obtained in closed form [7].

ONPP imposes that each low-dimensional sample y_i is locally reconstructed from its neighbors using exactly the same weights w_{ij} determined above. More specifically, ONPP captures this intuition using the criterion

$$E_{\text{onpp}} = \sum_i \|y_i - \sum_j w_{ij} y_j\|_2^2. \tag{6}$$

Again it can be shown [3] that this criterion can be re-written as

$$\begin{aligned} E_{\text{onpp}} &= \text{Tr}\,[YMY^\top] \\ &= \text{Tr}\,[V^\top XMX^\top V], \end{aligned} \tag{7}$$

where we have used (1) and introduced the matrix $M := (I - W^\top)(I - W)$.

As for OLPP, the matrix V should have orthonormal columns, leading to the constrained optimization problem

$$\begin{cases} \min\limits_{V \in \mathbb{R}^{m\times d}} & \text{Tr}\,\left[V^\top XMX^\top V\right], \\ V^\top V = I \end{cases} \tag{8}$$

which is again a standard eigenvalue problem.

To summarize, both OLPP and ONPP solve an eigenvalue problem and they determine the matrix V from a few eigenvectors of the matrices $XLX^\top$ and $XMX^\top$, respectively.

3. Linear dimensionality reduction for evolutionary data

In this section, we generalize the above methods to deal with evolutionary data. We attach the subscript t in order to refer to a certain quantity at time step t. For example, X_t and V_t denote the data set and the dimensionality reduction matrix respectively, at time step t. We will first use OLPP as an illustration and then consider the extension to ONPP.

We would like to perform dimensionality reduction in a temporally smooth fashion, such that the learned subspace is robust to short-term changes and at the same time adaptive to long-term drifts in the data. One way to achieve this, is to introduce a penalty term in the OLPP objective function that discourages V_t from being too far from V_{t-1}. Previous work [5] employed the following penalty term:

$$\frac{1}{2}\|V_t V_t^\top - V_{t-1} V_{t-1}^\top\|_F^2, \tag{9}$$

which compares the two subspaces spanned by the columns of V_t and V_{t-1} and is invariant to the particular realizations of the bases V_t and V_{t-1}. Note that this defines a notion of distance between these subspaces, termed the *gap* when the 2-norm is used instead of the Frobenius norm, see e.g., [8, p. 86], and [9].

In the following, we discuss two ways of combining the penalty term (9) with the OLPP objective function (4). The first approach takes a fixed weighted linear combination of the two terms. The second approach takes the ratio between them.

3.1. Linear combination of the two objectives

Recall from (4) that the objective function of OLPP at time step t reads $\mathrm{Tr}\,[V_t^\top \tilde{L}_t V_t]$, where we introduced the matrix

$$\tilde{L}_t := X_t L_t X_t^\top. \tag{10}$$

Incorporating the penalty term (9) results in

$$(1-\beta)\ \mathrm{Tr}\ \left[V_t^\top \tilde{L}_t V_t\right] + \frac{\beta}{2}\|V_t V_t^\top - V_{t-1}V_{t-1}^\top\|_F^2, \tag{11}$$

where $\beta \in [0,1)$ is a free parameter that determines a balance between the two terms. Observe that $\|V_t V_t^\top - V_{t-1}V_{t-1}^\top\|_F^2$ can be re-written as follows

$$\begin{aligned}
&\mathrm{Tr}\ \left[(V_t V_t^\top - V_{t-1}V_{t-1}^\top)(V_t V_t^\top - V_{t-1}V_{t-1}^\top)\right] \\
&= \mathrm{Tr}\ \left[V_t V_t^\top V_t V_t^\top\right] + \mathrm{Tr}\ \left[V_{t-1}V_{t-1}^\top V_{t-1}V_{t-1}^\top\right] \\
&\quad - 2\mathrm{Tr}\ \left[V_t V_t^\top V_{t-1}V_{t-1}^\top\right] \\
&= 2d - 2\mathrm{Tr}\ \left[V_t^\top V_{t-1}V_{t-1}^\top V_t\right].
\end{aligned}$$

Hence, (11) can be simplified to

$$\begin{aligned}
&(1-\beta)\ \mathrm{Tr}\ \left[V_t^\top \tilde{L}_t V_t\right] - \beta\ \mathrm{Tr}\ \left[V_t^\top V_{t-1}V_{t-1}^\top V_t\right] \\
&=\ \mathrm{Tr}\ \left[V_t^\top((1-\beta)\tilde{L}_t - \beta V_{t-1}V_{t-1}^\top)V_t\right].
\end{aligned}$$

To summarize, the optimization problem for (11), called OLPP-E in the following, becomes

$$\begin{cases} \min\limits_{V_t \in \mathbb{R}^{m\times d}} & \mathrm{Tr}\ \left[V_t^\top((1-\beta)\tilde{L}_t - \beta V_{t-1}V_{t-1}^\top)V_t\right]. \\ V_t^\top V_t = I \end{cases} \tag{12}$$

Again, this turns out to be an eigenvalue problem and the solution is given by the matrix V_t containing an orthonormal basis of eigenvectors associated with the smallest d eigenvalues of $(1-\beta)X_t L_t X_t^\top - \beta V_{t-1}V_{t-1}^\top$.

REMARK **3.1.** *It is straightforward to obtain a similar formulation for ONPP. In this case, the corresponding optimization problem becomes*

$$\begin{cases} \min\limits_{V_t \in \mathbb{R}^{m\times d}} & Tr\ \left[V_t^\top((1-\beta)\tilde{M}_t - \beta V_{t-1}V_{t-1}^\top)V_t\right], \\ V_t^\top V_t = I \end{cases} \tag{13}$$

where $\tilde{M}_t = X_t M_t X_t^\top$.

3.2. Ratio of the two objectives

Instead of taking a linear combination of the two objectives, one could also take the ratio

$$\frac{\mathrm{Tr}\ \left[V_t^\top \tilde{L}_t V_t\right]}{\mathrm{Tr}\ \left[V_t^\top (V_{t-1}V_{t-1}^\top)V_t\right]}. \tag{14}$$

Such an approach avoids the need of choosing a parameter. Minimizing (14) aims at minimizing the numerator corresponding to the OLPP objective function and at the same time maximizing the denominator. Note that for the denominator term to be large, the two projectors must be close.

This choice of criterion for the evolutionary OLPP results in the optimization problem

$$\begin{cases} \min\limits_{V_t \in \mathbb{R}^{m\times d}} & \dfrac{\mathrm{Tr}\ \left[V_t^\top \tilde{L}_t V_t\right]}{\mathrm{Tr}\ \left[V_t^\top (V_{t-1}V_{t-1}^\top)V_t\right]}. \\ V_t^\top V_t = I \end{cases} \tag{15}$$

Unlike (12), this optimization problem is *not* equivalent to a standard eigenvalue problem and turns out to be much harder. More specifically, a problem of the same form has been considered in [10] and [11], where it has been shown that (15) is equivalent to minimizing the function

$$f(\gamma) := \min_{V\in\mathbb{R}^{n\times d},\ V^\top V = I} \mathrm{Tr}\,[V^\top(A-\gamma B)V], \tag{16}$$

with $A = \tilde{L}_t = X_t L_t X_t^\top$, and $B = V_{t-1}V_{t-1}^\top$.

For fixed γ, the optimal V will be denoted by $V(\gamma)$ and is an orthonormal basis of eigenvectors associated with the smallest d eigenvalues of $A-\gamma B$. Given the following two conditions,

$$\mathrm{Null}(A) \cap \mathrm{Null}(B) = \{0\}, \tag{17}$$

$$\mathrm{rank}(B) > m - d., \tag{18}$$

it was shown in [11] that f is a non-increasing function and has a unique zero γ_*, which amounts to the minimum of the trace ratio

$$\frac{\mathrm{Tr}\,[V^\top A V]}{\mathrm{Tr}\,[V^\top B V]}. \tag{19}$$

While condition (17) can be expected to be satisfied, it is rather unlikely that (18) is satisfied, as $\mathrm{rank}(B) = d$ and usually $m \gg d$. This problem can be circumvented by regularizing B, see also Section 4.

Consequently, the trace ratio optimization problem can be approached by solving $f(\gamma) = 0$ by using, e.g., the Newton method. For this purpose, the derivative of f is needed, which – according to [11] – is

$$f'(\gamma) = -\mathrm{Tr}\,[V(\gamma)^\top B V(\gamma)],$$

with $V(\gamma)$ defined above. Hence, the Newton method takes the form

$$\begin{aligned}
\gamma_{new} &= \gamma - \frac{\mathrm{Tr}\,[V(\gamma)^\top(A-\gamma B)V(\gamma)]}{-\mathrm{Tr}\,[V(\gamma)^\top B V(\gamma)]} \\
&= \frac{\mathrm{Tr}\,[V(\gamma)^\top A V(\gamma)]}{\mathrm{Tr}\,[V(\gamma)^\top B V(\gamma)]}.
\end{aligned} \tag{20}$$

As suggested in [11], Algorithm 3.1 employs the Lanczos algorithm [12] to determine $V(\gamma)$.

ALGORITHM **3.1.** *Newton-Lanczos algorithm for Trace Ratio minimization*

1. *Input:* A, B *and a dimension* d.
2. *Select initial* $m \times d$ *matrix* V *with orthonormal columns.*
3. *Compute* $\gamma = Tr[V^\top AV]/Tr[V^\top BV]$.
4. *Until convergence Do:*
5. *Call Lanczos algorithm to compute the* d *smallest eigenvalues of* $G(\gamma) = A - \gamma B$ *and an associated orthonormal basis of eigenvectors* $V = [v_1, \cdots, v_d]$.
6. *Compute* $\gamma = Tr[V^\top AV]/Tr[V^\top BV]$.
7. *EndDo*

Algorithm 3.1 usually requires just a few steps to converge. We will refer to this variant of OLLP for evolutionary data as OLPP-ITR. A similar variant of ONPP can be obtained by taking $A = \tilde{M}_t$ in the derivations above.

4. Experimental results

In this section, we provide experimental results to investigate the behavior of the proposed methods, both on synthetic as well as on real-world evolutionary data. More specifically, we consider the following methods:

- OLPP: perform OLPP independently at each time step.
- OLPP-N: naive evolutionary OLPP, i.e., perform OLPP on the data set $\tilde{X}_t = [X_t, X_{t-1}]$.
- OLPP-E: evolutionary OLPP; see Section 3.1. The use of OLPP-E requires to choose a value for the parameter β. We use an ad hoc choice of $\beta = 0.5$ in all experiments.
- OLPP-ITR: evolutionary OLPP with iterative optimization; see Section 3.2. To increase the robustness of OLPP-ITR, the matrices A and B appearing in Algorithm 3.1 are scaled to have unit trace and are regularized by adding a small number to the diagonal entries.

Following common practice, a preliminary PCA is employed before any of these methods to reduce the dimensionality of the data to $n - c$, where c is the number of classes. In all methods, we build the affinity graph in a supervised fashion, as described in Section 2. All methods use Gaussian weights (2) where the parameter σ is half the median of pairwise Euclidean distances of a random sample subset. In all experiments, the data is classified in the reduced space using nearest neighbor classification.

4.1. Synthetic evolutionary data

We generate synthetic evolutionary data of two classes. First, $n_1 = 500$ data samples are drawn from the standard normal distribution $\mathcal{N}(0, 1)$ and assigned to the first class, and then $n_2 = 500$ data samples are drawn from $\mathcal{N}(-2, 1)$ and assigned to the second class. The dimension m of the data is set to 18 and each data sample is normalized to unit length. The first 10 samples from each class are assigned to the training set and the rest are assigned to the test set.

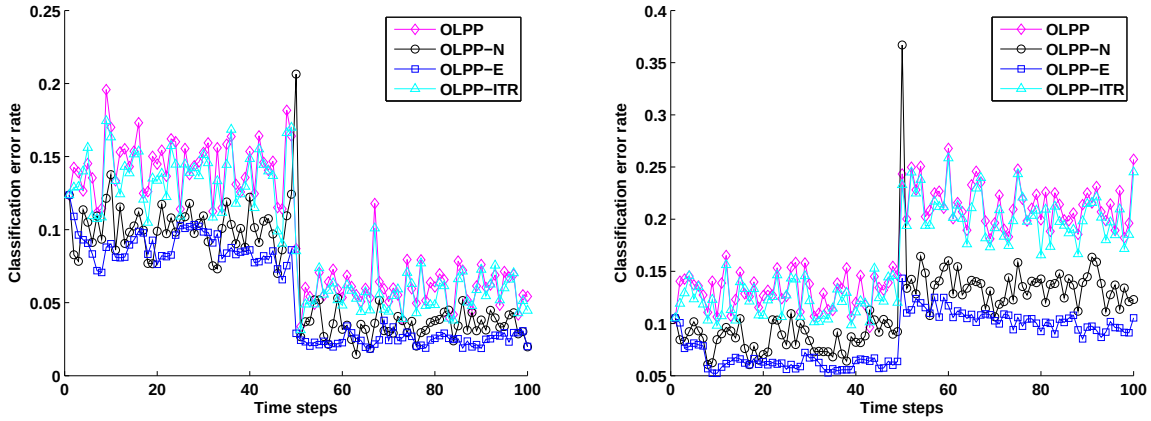
In order to simulate short-term noise, we add at every time step random perturbations – distributed according to $\mathcal{N}(0, 0.1)$ – to the training data. We generate data for 100 time steps. In order to simulate a concept drift, we shift the samples of the first class at time step $t_0 = 50$ by Δx. When $\Delta x > 0$, classification becomes easier as the first class moves away from the other. On the contrary, when $\Delta x < 0$, classification becomes harder since the first class moves closer to the other, increasing the overlap between the two classes. The generated data samples are normalized to unit length in order to remove any difference in scale among different time steps. The data are reduced to dimension $d = 3$.

We performed 20 random experiments corresponding to different random realizations of the data set. Figure 2 shows the average classification performance. Notice the effect of the concept drift around time step $t_0 = 50$, which makes the problem easier (left panel) or harder (right panel), respectively. Figure 2 clearly demonstrates that taking historic data into account is helpful for classification, as all evolutionary methods outperform OLPP applied independently at each time step. Notice also that the evolutionary methods are able to adapt quickly to the concept drifts.

Furthermore, OLPP-E outperforms the naive OLPP-N, since the latter is only using data from the time steps t and $t - 1$. In contrast, OLPP-E obtains weak supervision information from *all* previous time steps via V_{t-1}. In most cases, OLPP-ITR resulted in rather small values of β^* (of the order of 10^{-3}) which explains its performance close to OLPP independently at each time step. This is consistent with the findings in [11].

4.2. Evolutionary text data

We used a web crawler to collect paper abstracts from six journals over three main subject classes: physics, mathematics and biology. The



(a) $\Delta x = 0.1$ (b) $\Delta x = -0.1$

Figure 2: Average classification performance over time for the synthetic data corresponding two different concept drifts at time step $t_0 = 50$.

Method	AUC
OLPP	4.67
OLPP-N	3.11
OLPP-E	2.87
OLPP-ITR	4.17

Table 1: Classification performance (in terms of AUC) for the text data set.

selected journals have twelve monthly issues per year and the data collection period goes from 2003 to 2006. This results in an evolutionary text data set of $T = 4 \cdot 12 = 48$ time steps in total, where each month corresponds to a time step. Each paper abstract gives rise to a document whose class label is the corresponding subject class, i.e., physics, mathematics and biology. Hence, we have a multi-class classification problem with three classes in this data set. After data collection, we used the TMG tool[3] by [13, 14] to form the corresponding term-by-document matrix. The resulting matrix is of size 14918×3223, where 14918 is the number of terms and 3223 is the number of documents. We used term frequency weighting, i.e., the entry (i, j) of the term-by-document matrix stores the number of occurrences of the ith term in the jth document.

At each time step, we use five training samples per class and the rest of the samples are assigned to the test set. We set the dimension of reduced space to $d = 5$ for all methods. We measure the classification performance over 100 random realizations of the training and test sets.

[3]http://scgroup20.ceid.upatras.gr:8000/tmg/

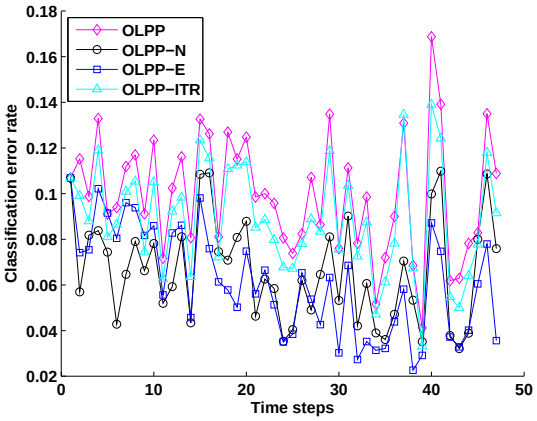
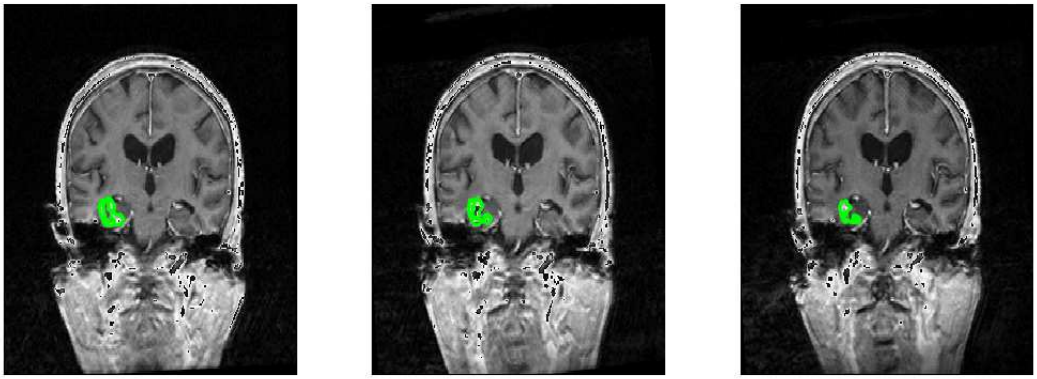
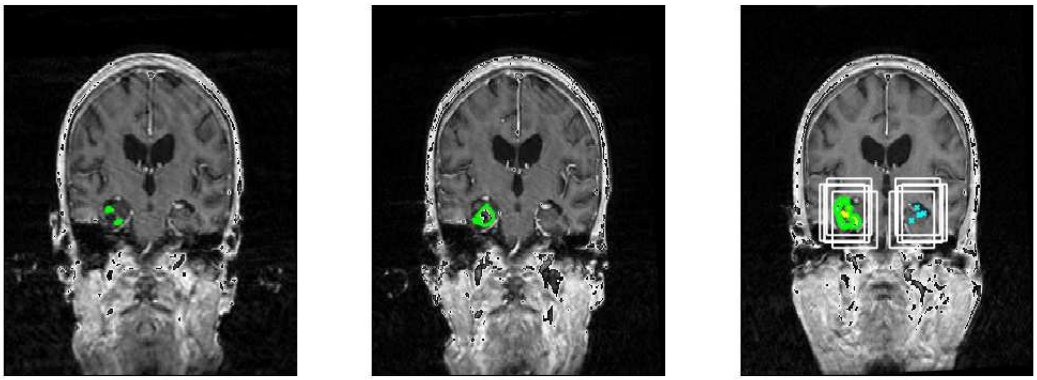


Figure 3: Classification performance over time for the text evolutionary data.

Figure 3 shows the average performance across time and Table 1 shows the performance in terms of AUC, which is the area under the classification error rate curve. AUC is an aggregate metric that summarizes the performance of each method across time and ideally it should be small. Observe that the conclusions from the results are similar to those of the synthetic case. Again, the use of historic data has a positive influence, letting the evolutionary methods outperform OLPP. Additionally, OLPP-E outperforms OLPP-N in all time steps after some initial transition phase. OLPP-ITR again computed small values for β^* (with a median value about 0.2), which resulted in a slightly better performance compared to the previous data set but is still inferior to the other evolutionary methods.



(a) $t = 1$ (b) $t = 2$ (c) $t = 3$



(d) $t = 4$ (e) $t = 5$ (f)

Figure 4: MRI brain image data. (a)-(e): fixed coronal level over time and corresponding tumor annotation; (f): sample random patches extracted from tumor (left) and healthy (right) region.

4.3. Brain image data

We use a data set that is publicly available from the Internet Brain Segmentation Repository (IBSR)[4]. The dataset contains five scans of a patient with a brain tumor taken at about 6 months intervals over three and a half years. Each image slice is of size 256×256. The scans are reasonably registered over time and they are annotated by a medical expert who provided the outline of the tumor region; see the first five panels in Figure 4 and the tumor outline colored in green. At each time step, we extract from each coronal slice several 40×40 patches randomly positioned around the tumor region, as well as in the corresponding healthy region from each slice (see Figure 4(f)). Hence, we have a patch classification problem on evolutionary data with two classes: healthy and tumor.

At each time step, we use 8 training samples per class and the rest of the samples are assigned to the test set. We set the dimension of reduced space to $d = 10$ for all methods. We work with a fixed coronal slice along time. For each slice, we measure the classification performance over 100 random realizations of the training and test sets. Table 2 shows the performance in terms of AUC, which is the area under the average classification error rate curve, where the average is computed across slices. Unfortunately, due to the fact that there are only five time steps in the data set, we cannot explore the behavior of the methods over large intervals in time. However, even from these few snapshots, one can see that taking historic data into account is helpful.

[4] www.cma.mgh.harvard.edu/ibsr/

Method	AUC
OLPP	0.94
OLPP-N	0.92
OLPP-E	0.85
OLPP-ITR	0.94

Table 2: Classification performance (in terms of AUC) in the brain image data set.

5. Conclusion

We have introduced and compared a few linear dimensionality reduction methods for the situation of evolutionary data. Although the discussion was centered around OLPP, it should be emphasized that similar evolutionary approaches can be developed for other reduction methods, such as ONPP, PCA, LDA, and so on. The experiments provide evidence that taking historic data into account by forcing the reduced space V_t to evolve in a temporally smooth fashion leads to improved classification performance.

References

[1] A. Webb, Statistical Pattern Recognition, 2nd edn, Wiley, 2002.

[2] X. He, P. Niyogi, Locality Preserving Projections, Advances in Neural Information Processing Systems 16 (NIPS) .

[3] E. Kokiopoulou, Y. Saad, Orthogonal Neighborhood Preserving Projections: A projection-based dimensionality reduction technique, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (12) (2007) 2143–2156.

[4] E. Kokiopoulou, J. Chen, Y. Saad, Trace optimization and eigenproblems in dimension reduction methods, Numerical Linear Algebra with Applications In press.

[5] Y. Chi, X. Song, D. Zhou, K. Hino, B. L. Tseng, Evolutionary spectral clustering by incorporating temporal smoothness, KDD (2007) 153–162.

[6] Y. Jia, S. Yan, C. Zhang, Semi-supervised classification on evolutionary data, International Joint Conferences on Artificial Intelligence (IJCAI) (2009) 1083–1088.

[7] S. Roweis, L. Saul, Nonlinear Dimensionality Reduction by Locally Linear Embedding, Science 290 (2000) 2323–2326.

[8] F. Chatelin, Eigenvalues of matrices, John Wiley & Sons Ltd., Chichester, 1993.

[9] Y. Saad, Numerical Methods for Large Eigenvalue Problems: Theory and Algorithms, John Wiley, New York, 1992.

[10] E. Kokiopoulou, Y. Saad, Enhanced graph-based dimensionality reduction with repulsion Laplaceans, Pattern Recognition 42 (11) (2009) 2392–2402.

[11] T. T. Ngo, M. Bellalij, Y. Saad, The trace ratio optimization problem, Tech. Rep., UMN, 2009.

[12] G. H. Golub, C. V. Loan, Matrix Computations, 3rd edn, The John Hopkins University Press, Baltimore, 1996.

[13] D. Zeimpekis, E. Gallopoulos, Design of a MATLAB toolbox for term-document matrix generation, In Proc. Workshop on Clustering High Dimensional Data and its Applications, (held in conjunction with 5th SIAM Int'l Conf. Data Mining) (2005) 38–48.

[14] D. Zeimpekis, E. Gallopoulos, TMG: A MATLAB toolbox for generating term document matrices from text collections, Grouping Multidimensional Data: Recent Advances in Clustering, Springer (2006) 187–210.