Journal of Computer-Aided Molecular Design **18:** 475–482, 2004.
DOI 10.1007/s10822-004-5319-9

Locally Linear Embedding for dimensionality reduction in QSAR

P.-J. L'Heureux[a,*], J. Carreau[a], Y. Bengio[a], O. Delalleau[a] & S.Y. Yue[b]
[a]*DIRO, Université de Montréal, C.P. 6128, Succ. Centre-Ville, Montréal, Canada H3C 3J7;* [b]*AstraZeneca R&D Montreal, Canada*

Received 4 May 2004; accepted in revised form 18 October 2004

Key words: kernel methods, neural network, QSAR, spectral dimensionality reduction

Summary

Current practice in Quantitative Structure Activity Relationship (QSAR) methods usually involves generating a great number of chemical descriptors and then cutting them back with variable selection techniques. Variable selection is an effective method to reduce the dimensionality but may discard some valuable information. This paper introduces Locally Linear Embedding (LLE), a local non-linear dimensionality reduction technique, that can statistically discover a low-dimensional representation of the chemical data. LLE is shown to create more stable representations than other non-linear dimensionality reduction algorithms, and to be capable of capturing non-linearity in chemical data.

Introduction

Quantitative Structure Activity Relationship (QSAR) is a set of methods that try to find a statistical relationship between a set of descriptors of compounds and their activity [1]. Unfortunately, our physico-chemical understanding of the reactions involved is often insufficient to make the most appropriate choice of descriptors from purely chemical considerations. Hopefully, if we acquire enough data, statistics can help in the choice of descriptors.

The generalization of classical QSAR models has been very much limited by a low number of systematically generated data. As combinatorial chemistry can screen hundreds of molecules, it has created an opportunity to use neural networks and other powerful statistical algorithms to produce the statistical model. Our hypothesis is that such learning algorithms can be sufficiently supplied with data to beat traditional regression analysis.

A serious problem for many statistical modeling approaches is the high dimension in which the chemical data lie. The current trend for reducing the dimensionality involves variable selection. Two areas of concern exist about variable selection. First, all those methods are based on the assumption that most descriptors are irrelevant, thus forgoing the information contained in the discarded descriptors. Second, the interdependences of descriptors are not used at their full potential. If there exists interdependence between descriptors, then structure will arise in the data. Dimensionality reduction aims to capture this structure, which can be viewed as a surface in high-dimension. The end result of non-linear dimensionality reduction is a transformation of the coordinate system to project the data onto this low-dimensional surface. Dimensionality reduction (especially non-linear) thus has the potential to preserve the information in the data while representing it more compactly.

We investigate the use of Locally Linear Embedding (LLE) [2] as a dimensionality reduction procedure before applying Partial Least Squares or neural networks. Reducing the dimension of the descriptor space might help the super-

*To whom correspondence should be addressed. Tel: +1-514-343-6111, ext: 1794; E-mail: lheureup@iro.umontreal.ca

vised learning algorithms to learn a better model from the data and to avoid overfitting. We also present a method that allows to circumvent one of the limitations of LLE, namely that it requires both the training data and the test data in order to infer a low-dimensional representation for the test data. The new method allows to apply LLE – as well as other similar spectral dimensionality reduction methods – to new (out-of-sample) examples after it has been trained on a training set. This allows to introduce a novel form of analysis for dimensionality reduction methods, based on the stability of the induced embedding, and compare three spectral dimensionality reduction methods on that basis (LLE, Isomap, and kernel PCA). With this analysis, we assess how the complexity of the chemical data is handled by LLE, as we feed it with more data.

Machine learning toolkit

We will describe here the basic machine learning concepts used in our experiments. We first give a description of what are the goals and techniques of dimensionality reduction, and in particular spectral methods such as LLE. We then describe the neural network and Partial Least-Squares (PLS) model which are used for regression. We finally explain how we used the double cross-validation procedure.

Dimensionality reduction

When the data lie in a high-dimensional space, it might be useful to look for a more compact representation. If it is possible to express the main types of variations in the data with a small set of numbers then fewer parameters need to be learned in order to make predictions from these numbers. Some dimensions might just be meaningless or redundant, and some might be sufficient to carry the most important variations found in the data. To achieve this, one simply looks for a low-dimensional surface (more generally, manifold) near which most of the data lie. The simplest approach assumes that the manifold is a hyper-plane. A standard algorithm for this purpose is PCA, 'Principal Component Analysis' [3]. PCA computes the top eigenvectors of the data covariance matrix. Those vectors correspond to the linear projections of greatest variance.

Local non-linear dimensionality reduction

PCA assumes that the low-dimensional embedding is obtained as a linear projection of the data. This assumption might not be justified and several algorithms were designed to perform non-linear dimensionality reduction. Among them, recent ones are Isomap [4] and LLE [2, 5]. In this project we focus on LLE, but perform comparative experiments with Isomap and kernel PCA [6], which are related to LLE: all three are non-parametric methods based on 'spectral embedding', as discussed in [7].

The main assumption involved in all three methods is that the data are lying near a smooth manifold of lower dimension and that this manifold can be approximated locally by linear patches. These patches are constructed around each training point. With LLE, one associates with each training example (and corresponding linear patch) a linear combination of its nearest neighbors which best reconstructs the data point. The assumption here is that the same linear combination of neighbors can reconstruct the point in a lower dimension, i.e. on the manifold, and this would be true if the neighbors are close enough compared to the curvature of the manifold, i.e. one can locally approximate the manifold by a linear surface near which the example and its neighbors lie. Therefore, we have two parameters that control the projection of the data points in a space of lower dimension: the dimension of the manifold near which the data are assumed to lie and the number of neighbors used to reconstruct the linear patches (which indirectly controls the size of the patches). The memory requirement of our current implementation grows as $O(k \times n)$, where n is the size of the training set, and k is the number of neighbors. The processing time will scale as $O(n^2 \times d)$, where d is the manifold dimension.

Out-of-sample extension for locally linear embedding

In its original formulation [5], LLE only provides an embedding for its training set. When a new point x is given and its low-dimensional coordinates are needed, retraining LLE with x in the training set would be too costly if it had to be done for each new example. Instead, it is proposed in [2] to compute the reconstruction of x from its k nearest neighbors $x_{i_1}, \ldots, x_{i_k}$, and to find the low-dimensional point $P(x)$ using the neighbors' low-

dimensional coordinates $P(x_{i_1}), \ldots, P(x_{i_k})$. More formally, let $(W_j)_{1 \leq j \leq k}$ be the neighbors' weights for the reconstruction of x, i.e., such that $\sum W_j = 1$ and $||x - \sum W_j x_{i_j}||^2$ is minimized. The low-dimensional coordinates $P(x)$ are obtained from those of the neighbors by $P(x) = \sum W_j P(x_{i_j})$.

This formula is justified intuitively by the local linear assumption in LLE. In addition, it has been shown in [7] that there is a more general justification to this extension method, that applies for a larger family of unsupervised learning algorithms called spectral embedding algorithms. These algorithms perform dimensionality reduction using a **kernel function** (a symmetric function $K(x, y)$ applied on pairs of input points). All of these algorithms compute the $(n \times n)$ Gram matrix M defined by $M_{ij} = K(x_i, x_j)$, with $x_1, \ldots, x_n$ the training set, and obtain the low-dimensional coordinates of a point x_i by $P(x_i) = (v_{1i}, v_{2i}, \ldots, v_{di})'$, where $v_r = (v_{r1}, \ldots, v_{rn})'$ is the r-th principal eigenvector of M (an additional scaling may apply, depending on the algorithm, e.g. in LLE this embedding is multiplied by $\sqrt{n}$). Let $P_r(x)$ be the r-th low-dimensional coordinate on the manifold of a point x. If $x = x_i$ is in the training set, we thus have $P_r(x_i) = v_{ri}$ (up to scaling). If x is not in the training set, we can 'extrapolate' the eigenvectors by the Nyström formula (which has been used previously for estimating extensions of eigenvectors in Gaussian process regression [8]). This formula is:

$$P_r(x) = \frac{1}{\ell_r} \sum_{i=1}^{n} P_r(x_i) K(x, x_i)$$

where ℓ_r is the eigenvalue associated to the eigenvector v_r. As shown in [7], there exists a kernel K such that this formula is equivalent to the extension of LLE [2] described previously. This interpretation gives additional insight into the stability properties of the embedding. Indeed, it links LLE with another popular spectral embedding algorithm, kernel PCA [6], that has been shown to be a convergent and stable algorithm [9–11]. Although there are some important differences between the two algorithms (in particular, the kernel used in LLE depends much more on the training data than the one in kernel PCA), these convergence properties suggest the LLE out-of-sample extension should also be stable provided enough training points are available: this stability has been verified empirically by experiments described in [7].

In this paper we have performed perturbation analysis with three different spectral dimensionality reduction methods: LLE (discussed above), Isomap [4], and kernel PCA [6]. Isomap [4] generalizes metric multi-dimensional scaling (MDS) [12] to non-linear manifolds. It is based on replacing the Euclidean distance by an empirical approximation of the geodesic distance $\mathscr{D}$ on the manifold (it is obtained as the shortest path in a graph whose nodes are the examples and only near neighbors are connected, with arc lengths equal to Euclidean distance). The Isomap algorithm obtains the normalized matrix M from which the embedding is derived by transforming the raw pairwise distances matrix as follows: (1) compute the matrix $\tilde{M}_{ij} = \mathscr{D}^2(x_i, x_j)$ of squared geodesic distances with respect to the data D and (2) apply to this matrix the distance-to-dot-product transformation

$$M_{ij} = -\frac{1}{2}(\tilde{M}_{ij} - \frac{1}{n}\sum_{j'} \tilde{M}_{ij'} - \frac{1}{n}\sum_{i'} \tilde{M}_{i'j} + \frac{1}{n^2}\sum_{i',j'} \tilde{M}_{i'j'})$$

as for MDS.

Kernel PCA generalizes the Principal Component Analysis approach to non-linear transformations using the kernel trick [6, 13, 14]. The matrix M is obtained by computing the above matrix M as follows:

$$M_{ij} = K(x_i, x_j) - \frac{1}{n}\sum_{j'} K(x_i, x_{j'}) - \frac{1}{n}\sum_{i'} K(x_{i'}, x_j) + \frac{1}{n^2}\sum_{i',j'} K(x_{i'}, x_{j'}).$$

Like for LLE and Isomap, the eigenvectors of M give the embedding of the training examples, and as shown in [7] they can be extended to provide the embedding of a test point.

Neural networks

There are many variants of multilayer neural networks and we describe here the most common [15], which was used in the experiments. We also limit our discussion to regression problems, in which the goal is to estimate a conditional expectation $E[A|\mathbf{x}]$ where A is a random variable that represents activity (e.g. log-concentration) and $\mathbf{x} = [x_1, \ldots, x_d]$ is the *input vector*, a fixed-size sequence of d

descriptor values. A one-hidden-layer neural network learns a function $f(\mathbf{x})$ of the form

$$f(\mathbf{x}) = b + \sum_{i=1}^{h} w_i \tanh(c_i + \sum_{j=1}^{d} v_{i,j} x_j)$$

with free parameters $\theta = (b, w_1, \ldots, w_h, c_1, \ldots, c_h, v_{1,1}, \ldots, v_{h,d})$, where h is the number of neurons in the hidden layer. Those parameters are optimized (by conjugate gradient descent) so as to minimize the mean squared error (MSE) on the training set, to which is added a regularization term called weight decay (proportional to the squared norm of the parameters).

Partial least-squares

PLS [16] was used as a benchmark against which we measured the performance of the neural network models. PLS is a linear statistical model which is frequently used in chemometrics.

The central assumption is that groups of observed variables are linearly related to groups of latent variables. Usually, we assume that the number of latent variables is smaller than the number of observed variables. This can be viewed as a form of dimensionality reduction. The particularity of this algorithm with respect to ordinary least-squares (OLS) is that it tries to simultaneously predict the target (the Y variable) and find a reduced representation of the input (the X variable) by modeling their cross-covariance $\mathrm{COV}(X, Y)$.

We used what is called two-blocks PLS. The software comes from a toolbox provided by Rasmus Bro and Claus A. Andersson called *The N-way toolbox for Matlab* at the website http://www.models.kvl.dk/source/nwaytoolbox/ index.asp.

Performance evaluation and hyper-parameters selection

The *hyper-parameters* of the neural networks are the number of hidden units and the weight-decay parameter (the coefficient that multiplies the squared norm of the parameters in the training criterion). Because of the reduced size of our datasets, they cannot be split into three disjoint sets (training, validation and test) large enough to allow meaningful training and error estimation. Basic cross-validation works this way: the data set is split into k folds which are used as follows. One set, let us say the ith set, is kept as the test set, the other $k-1$ sets are used as the training set. Each combination of training and test sets is considered as i varies from 1 to k. The error is estimated as the average error over the k test sets considered. In **double cross-validation**, a similar process is used to perform model selection within each of the outer folds: the $k-1$ sets forming the 'training plus validation' set are in turn split in l folds (the *inner* cross-validation), to estimate the error of each set of hyper-parameters and select a value for them. Note that the choice of hyper-parameter values may be different for each of the outer folds, so in the end we do not estimate the error of a particular choice of hyper-parameter value, but rather the error of the process which maps a data set into a function and that includes hyper-parameter selection. Thus we used double cross-validation to estimate the error **and** perform model selection.

Explored parameters for LLE, PLS and the neural network

We used a double five-by-five-fold cross-validation to choose the hyper-parameters (inner cross-validation) and evaluate generalization (outer cross-validation) The hyper-parameters for this model are grouped as follows:

1. The hyper-parameters for LLE are k, the number of nearest neighbors used to reconstruct the manifold of lower dimension and d, the dimension in which the data is projected. The values that were explored for these hyper-parameters are:

   k : 8 16 32
   d : 8 16 32 64 128

2. For PLS, the only hyper-parameter is the number of latent variables in the PLS model.

   latent variables : 1 2 ... 10

3. The hyper-parameters for the neural networks are h, the number of hidden units and λ, which controls the weight-decay penalty. The values that were explored for these hyper-parameters are:

   h : 1 2 4
   λ : 0.001 0.01 0.1 1 10 100

In some experiments, on top of the hyper-parameters explored for LLE, we also add the possibility

of using directly the data without projecting it at all.

Data

In this section, we describe the datasets used in this work.

Raw data sets

The first dataset is named *Benzodiazepines*. It is freely available at the QSAR Society website `http://www.ndsu.nodak.edu/qsarsoc/`. It originally contains 245 benzodiazepine compounds that act on the benzodiazepine receptor [17, 18]. After virtual screening of the dataset, we found two identical molecules that had differing biological activity. We deleted the lowest entry in both cases, namely: Ro-14-1359 and R0-13-9868.

Different benzodiazepines are used for different indications including: muscle relaxation, anxiety relief, treatment of convulsive disorders and specific types of seizures, sedation, insomnia, anesthesia (induction and maintenance), ethanol withdrawal.

The second dataset is named *Muscarinic* and contains 162 compounds that act on the M1 muscarinic receptor. It is freely available from Milano Chemometrics [18, 19].

The M1 muscarinic receptor is a G-protein coupled receptor. Of the five muscarinic acetylcholine receptor genes identifed to date, muscarinic m1, m3 and m5 receptors are thought to couple predominantly to the activation of phosphoinositidase C via the $G_{\frac{q}{11}}$ family of G-proteins. However, m1 and m3 muscarinic receptor-mediated stimulation of adenylate cyclase activity has also been observed in some cell lines.

Of the five muscarinic receptors, M1 is also the most densely distributed muscarinic receptor in the hippocampus and forebrain. It is believed to play a significant role in memory, specifically in processes for which the cortex and hippocampus interact.

The first two datasets have a continuous response for the $\log(\mathrm{IC}_{50})$.

The third dataset is named *AZCombiChem*, and contains 26000 compounds. They are a subset of the AstraZeneca R&D Montreal combinatorial chemistry library. Two percent of the compounds are actives. The dataset is proprietary.

Molecular descriptor calculation

For the datasets tested here, we were only given the molecular connectivities and the biological activities. Before computing any descriptors, we created a 3D representation, filtering out counterions, putting hydrogen atoms to obtain a neutral pH, calculating partial charge on each atom, and finally minimizing the free energy.

We have done all the descriptor calculations inside the MOE framework [20]. The important details are that each entry has a single molecule, that the molecular mechanics force field is MMFF94 and the partial charge uses a bond charge increment method [21].

All the descriptors for each compound were computationally produced. We do not have experimental values in our dataset, apart from the activity measurements. The software used to compute the descriptors was MOE from CCG. MOE produces 474 descriptors from 2D to 3D representations. The two public datasets processed with MOE can be obtained upon request.

Assessment of unsupervised learning: size of training set needed for learning

The out-of-sample extension of LLE enables us to construct a test of convergence that does not depend on the activity data. The convergence test can be used towards answering some fundamental questions about the dimensionality reduction step. One of these questions is which size of the training set is needed to construct a stable embedding. Obtaining a stable embedding would mean that the reduced space would correspond to a generalized chemical manifold. The question is fundamental because it relates to the structure of the data in high dimension. The hunger for data of any non-linear dimensionality reduction technique will grow with the complexity of the data structure. It is thus important to assess how the complexity of the chemical data is handled by LLE and alternative methods.

In order to qualitatively determine the size of the training set needed to obtain a stable embedding, we devised the following test. In a large database of compounds, we first put aside a test set. We then select randomly two equal size and disjoint data subsets. We individually

compute the embedding on each subset. We then project the test set into the two reduced spaces. After an affine transformation that aligns the projected test sets, we compute the perturbation error, which is the average squared difference between the affine-aligned embeddings. For statistical significance, we repeat 30 times for each size of the data subsets. We tried many training set sizes on the *AZCombiChem* data set. We tested four dimensionality reduction methods: Isomap, Additive Kernel PCA, Divisive Kernel PCA and LLE. Because the perturbation error depends on the number of principal components, only the shape of the curve is meaningful across different methods.

Figure 1 shows clearly that LLE continues to improve its stability as the size of the training set increases. This could mean two things. Either LLE had an unstable embedding to begin with and needs a lot more data than other dimensionality reduction techniques, or LLE succeeds in capturing more of the complexity of the chemical data.

Note that the capacity to create a stable kernel and capture structure in the data is different from capturing the relevant information in the data. Ultimately, a good measure of the capacity to capture relevant information is through supervised learning.

Results of dimensionality reduction on supervised learning

The space where the descriptors of the molecules lie has a very high dimension (474). Moreover, the training set size is quite small (a few hundreds of examples). High dimensionality and small training set size together make the learning task very difficult. We will try to demonstrate the usefulness of LLE to tackle such a problem.

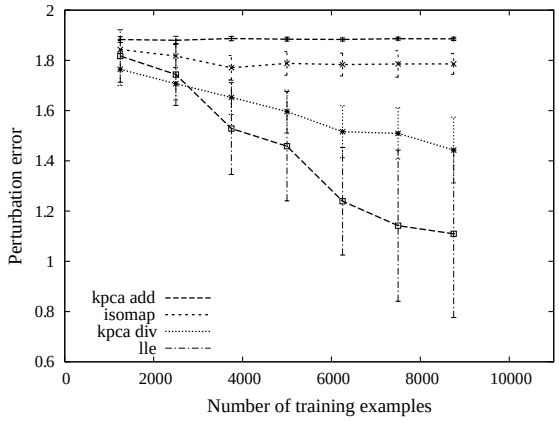

Figure 1. Stability of different non-linear dimensionality reduction techniques when varying the size of the training set. AZCombiChem dataset with MOE descriptors.

Capturing non-linearity

LLE, like other dimensionality reduction techniques, can be viewed as distorting the data. Starting with a very complicated dataset, it tries to find some locally linear manifold in the data. The neural network thus works on a less complicated space, from which a linear relationship with activity may be easier to capture. To support our conjecture, we tried a single unit neural network without LLE preprocessing.

Specific results for these datasets with the neural network models show that often only one hidden unit is kept in the hidden layer during cross-validation. This could be explained by the small amount of data and the amount of noise (lack of predictability from the given input descriptors), or by a truly simple relation between inputs and outputs (which is unlikely). Note that if the compounds are close together in space, the manifold in which they lie may be simpler.

In order to characterize the learning of non-linearity, we designed two experiments with the *Benzodiazepines* dataset. The first one is designed to show whether LLE is capable of learning non-linearity in the chemical data. The second is designed to show whether the nonlinearity in the neural network is useful. Note that for both experiments, LLE usage is a choice inside the double cross-validation. Both experiments use two neural networks constrained to have one hidden

Table 1. Single unit neural network with different activation functions on the *Benzodiazepines* dataset, with or without LLE preprocessing.

Activating function	Hidden units	LLE	MSE	Spread	Time (h)
Linear	1	No	5.50	2.3	3
Linear	1	Yes	0.49	0.05	6
tanh	1	No	0.62	0.07	1
tanh	1	Yes	0.58	0.06	3
tanh	Many	Yes	0.51	0.05	43
Test/training size 48/195					

Table 2. Predictive model performance on *Benzodiazepines* and *Muscarinic* datasets with or without LLE.

Dataset	Model	LLE	MSE	Spread	Test/training size
Benzodiazepines	PLS	No	0.82	0.08	48/195
Benzodiazepines	PLS	Yes	0.75	0.08	48/195
Benzodiazepines	NNet	No	0.71	0.09	48/195
Benzodiazepines	NNet	Yes	0.52	0.05	48/195
Muscarinic	PLS	No	0.68	0.08	32/130
Muscarinic	PLS	Yes	0.68	0.08	32/130
Muscarinic	Nnet	No	0.67	0.12	32/130
Muscarinic	NNet	Yes	0.49	0.06	32/130

unit. The first network has a linear activation function and the second one uses tanh.

The mean squared error (MSE) is estimated by double cross-validation, as explained previously. The spread is an estimator of standard deviation of the MSE. Computation time is on a linux workstation with a 2GHz P4 CPU, and includes the total time of the double cross-validation experiment (training $90 \times 5 \times 5$ times when there is a single hidden unit, 3 times more with 3 choices of hidden layer size). Note that LLE in itself is quick.

Table 1 summarizes the results of the two experiments. For the first experiment, the results of the two linear networks show that LLE alone can learn the nonlinearity in the data. For the second experiment, if we compare the linear and tanh network without LLE, we see that the tanh function captures the nonlinearity in the data.

In fact, choosing the number of hidden units by cross-validation or forcing it to be one has small effect on the results. This can be seen in Table 1, where no matter the method of hidden unit selection, we get around 0.5. It would thus seem that the neural network treats the relationship between chemical descriptors and biological activity as linear, after LLE preprocessing. Indeed, Table 1 shows that one should either use LLE or the nonlinear activation function in order to obtain good results with those special neural networks.

Learning with LLE

We now present learning experiments where we test the usefulness of LLE as a preprocessing to PLS or neural networks learning.

The results for the *Benzodiazepines* and *Muscarinic* datasets are summarized in Table 2. Note that for every experiment where LLE was a choice inside the double cross-validation, it has been chosen as the best model.

Table 2 first shows that neural networks always give better results than PLS, for a given choice of LLE. Note that the spread of the data is important, so the difference between differing models cannot be statistically proved. In particular, Table 1 shows better results with a linear model than with a full neural network.

Table 2 also shows that LLE almost always enhances the learning. Either PLS or neural network models get better predictive power after LLE preprocessing. The only exception is with PLS on the *Muscarinic* dataset, where we see no enhancement.

Conclusions

In this work, we have described an experimental investigation of LLE in QSAR. We have also shown how to use LLE to obtain a low-dimensional representation for new compounds.

We introduced a novel form of analysis for dimensionality reduction methods, based on stability of the induced embedding, and compared three spectral dimensionality reduction methods on that basis (LLE, Isomap, and kernel PCA). We conclude that the power of LLE to capture the chemical manifold will grow as we enlarge the training set, and this power appears greater than that of other non-linear non-parametric dimensionality reduction methods (Isomap and kernel PCA).

We then tried specially designed neural networks to investigate the capacity of LLE to

capture nonlinearity. We conclude that LLE can capture as well as a neural network the nonlinearity in the relatively small datasets tested here.

Finally, we ran several experiments to demonstrate the enhancement of LLE to the predictive power of models built with PLS and neural networks. For the datasets at hand, the combination of LLE and PLS or LLE and a neural network yields better average test error.

Other investigations should be done with more and larger datasets to improve the statistical significance of the enhancement provided by LLE. Nonetheless, the current results show that LLE's low-dimensional representations will increase in stability as we increase the training set size. Future work should also focus on comparing LLE with variable selection techniques.

Acknowledgements

We thank *Valorisation Recherche Québec* and the NSERC for financial support.

References

1. Hansch, C. and Leo, A., Exploring QSAR: Fundamentals and Applications in Chemistry and Biology. ACS Professional Reference Book, 1995.
2. Saul, L. and Roweis, S., J. Mach. Learn. Res., 4 (2002) 119.
3. Jolliffe, I.T., Principal Component Analysis. Springer, New York, 2002.
4. Tenenbaum, J., de Silva, V. and Langford, J., Science, 290(5500) (2000) 2319.
5. Roweis, S. and Saul, L., Science, 290(5500) (2000) 2323.
6. Schölkopf, B., Smola, A. and Müller, K.-R., Neural Comput., 10 (1998) 1299.
7. Bengio, Y., Paiement, J., Vincent, P., Delalleau, O., Le Roux, N. and Ouimet M., Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. In Thrun, S., Saul, L. and Schölkopf, B. (Eds), Advances in Neural Information Processing Systems 16, 2004.
8. Williams, C.K.I. and Seeger, M., Using the Nyström method to speed up kernel machines. In Leen, T., Dietterich, T. and Tresp, V. (Eds), Advances in Neural Information Processing Systems 13. Cambridge, MA, 2001, pp. 682–688.
9. Shawe-Taylor, J., Cristianini, N. and Kandola., J., On the concentration of spectral properties. In Dietterich, T., Becker, S. and Ghahramani, Z. (Eds), Advances in Neural Information Processing Systems 14, 2002.
10. Shawe-Taylor, J. and Williams, C., The stability of kernel principal components analysis and its relation to the process eigenspectrum. In Becker, S., Thrun, S. and Obermayer, K. (Eds), Advances in Neural Information Processing Systems 15, 2003.
11. Zwald, L., Bousquet, O. and Blanchard, G., Statistical Properties of Kernel Principal Component Analysis. In Shawe-Taylor, J. and Singer, Y. (Eds), Learning Theory: Proceedings of 17th Annual Conference on Learning Theory, COLT 2004, Banff, Canada, July 1–4. Vol. 3120 of Lecture Notes in Computer Science. Springer, Berlin, Germany, 2004, pp. 594–608.
12. Cox, T. and Cox, M., Multidimensional Scaling. Chapman & Hall, London, 1994.
13. Schölkopf, B., Smola, A. and Müller, K.-R., Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Technical Report 44, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, 1996.
14. Schölkopf, B., Burges, C.J.C. and Smola, A.J., Advances in Kernel Methods – Support Vector Learning, MIT Press, Cambridge, MA, 1999.
15. Rumelhart, D., Hinton, G. and Williams, R., Nature, 323 (1986) 533.
16. Frank, I. and Friedman, J., Technometrics 35(2) (1993) 109.
17. Harrison, P., Barlin, G., Davies, L., Ireland, S., Matyus, P. and Wong, M., Eur. J. Med. Chem., 31(1996) 651.
18. Burden, F., Ford, M., Whitley, D. and Winkler, D., J. Chem. Inf. Comput. Sci., 40 (2000) 1423.
19. Orlek, B., Blaney, F., Brown, F., Clark, M., Hadley, M., Hatcher, J., Riley, G., Rosenberg, H., Wadsworth, H. and Wyman, P., J. Med. Chem., 34 (1991) 2726.
20. Santavy, M. and Labute, P., SVL: The Scientific Vector Language, 1997. www.chemcomp.com/Journal_of_CCG/Features/svl.htm.
21. Halgren, T., J. Comput. Chem., 17 (1996) 490.