JMLR: Workshop and Conference Proceedings 29:1–16, 2013 ACML 2013

Learning a Metric Space for Neighbourhood Topology Estimation: Application to Manifold Learning

Karim T. Abou–Moustafa ABOUMOUS@CS.UALBERTA.CA
Dept. of Computing Science
University of Alberta
Edmonton, Alberta T6G 2E8, Canada

Dale Schuurmans DALE@CS.UALBERTA.CA
Dept. of Computing Science
University of Alberta
Edmonton, Alberta T6G 2E8, Canada

Frank Ferrie FERRIE@CIM.MCGILL.CA
Centre for Intelligent Machines
McGill University
Montréal, Quebec H3A 0E9, Canada

Editor: Cheng Soon Ong and Tu Bao Ho

Abstract

Manifold learning algorithms rely on a neighbourhood graph to provide an estimate of the data's local topology. Unfortunately, current methods for estimating local topology assume local Euclidean geometry and locally uniform data density, which often leads to poor data embeddings. We address these shortcomings by proposing a framework that combines local learning with parametric density estimation for local topology estimation. Given a data set $\mathcal{D} \subset \mathcal{X}$, we first estimate a new metric space $(\mathbb{X}, d_{\mathbb{X}})$ that characterizes the varying sample density of $\mathcal{X}$ in $\mathbb{X}$, then use $(\mathbb{X}, d_{\mathbb{X}})$ as a new (pilot) input space for the graph construction step of the manifold learning process. The proposed framework results in significantly improved embeddings, which we demonstrated objectively by assessing clustering accuracy.

Keywords: Manifold learning, neighbourhood graphs, neighbourhood topology, divergence based graphs, low rank covariance matrix estimation.

1. Introduction

Manifold learning algorithms have recently played a crucial role in unsupervised learning tasks such as clustering and nonlinear dimensionality reduction (Roweis and Saul, 2000; Tenenbaum et al., 2000; Ng et al., 2002; Brand, 2003; Belkin and Niyogi, 2003; Weinberger and Saul, 2004; Shaw and Jebara, 2009). Many such algorithms have been shown to be equivalent to Kernel PCA (KPCA) with data dependent kernels, itself equivalent to performing classical multidimensional scaling (cMDS) in a high dimensional feature space (Scholkopf et al., 1998; Williams, 2002; Bengio et al., 2004).

A common aspect of these algorithms is that they rely on a neighbourhood graph constructed from the input data $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n$ with the points $\mathbf{x}_i \in \mathbb{R}^d$ as its vertices.[1] Such a graph provides an estimate for the topology of the underlying low dimensional manifold that (approximately) encapsulates the data. A manifold learning algorithm then tries to "unfold", or "flatten" this manifold—while preserving some local information—to partition the graph (e.g. as in clustering), or to redefine some metric information (e.g. as in dimensionality reduction). The standard distance used to construct this graph—whether it be a fully connected, ϵ-ball, or k-nearest neighbours graph—is the Euclidean distance. Unfortunately, the Euclidean distance creates severe inaccuracy problems for graph estimation, and thus for the subsequent manifold learning process. In this paper, we show how to overcome these inaccuracies by introducing a new manifold learning framework that mitigates the liability incurred by using Euclidean geometry on real data.

One reason for inaccurate topology estimates is the finite nature of data, which means that low probability regions will be poorly sampled and hence poorly represented in the data $\mathcal{D}$. This results in an *uneven sample distribution* in the input space $\mathcal{X}$ (Bottou and Vapnik, 1992). In practice, the situation is exacerbated by the multimodal nature of the data, noise, nonlinearity, and its high dimensionality. Unfortunately, the Euclidean metric cannot accommodate any of these factors. First, the Euclidean distance, by definition, assumes a constant density over the entire input space $\mathcal{X}$, and hence does not take the varying sample distribution into consideration. To see this note that for any data of the form $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n$, the Euclidean distance enforces an identity covariance matrix $\mathbf{I}$ to measure pairwise distances between points. By expanding the squared norm of $\|\mathbf{x}-\mathbf{y}\|^2$ to $(\mathbf{x}-\mathbf{y})^\top\mathbf{I}(\mathbf{x}-\mathbf{y})$, one obtains a special case of the generalized quadratic distance (GQD) $d(\mathbf{x},\mathbf{y};\mathbf{A}) = \sqrt{(\mathbf{x}-\mathbf{y})^\top\mathbf{A}(\mathbf{x}-\mathbf{y})}$ which itself, generalizes the Mahalanobis[2] distance for any symmetric positive definite matrix $\mathbf{A}$. If $\mathbf{A} = \mathbf{I}$, then the Euclidean distance enforces a unit variance for all the variables in the data with zero correlation among them.

Second, similar remarks apply for the GQD if $\mathbf{A}$ is the inverse of the data's global covariance matrix, or it is learned via a metric learning algorithm (Xing et al., 2002; Weinberger et al., 2006) that might impose some local and/or global constraints on distances (based on labels or side information). For most metric learning algorithms $\mathbf{A}$ is constant over $\mathcal{X}$ and hence, it is still not a faithful modelling for the varying density in $\mathcal{X}$. More importantly, such metric learning algorithms are either supervised or semi-supervised, and hence they cannot be used in the unsupervised setting discussed here. These factors impart serious inaccuracy in the data graph construction, which in turn yields erroneous estimates for the manifold topology and increases uncertainty of point locations in the lower dimensional subspace. A manifestation of these effects is topological instability of manifold learning and sensitivity to noise (Balasubramanian et al., 2002).

Recently, it has been observed that the majority of manifold learning algorithms can be expressed as a regularized loss minimization of a reconstruction matrix, followed by a singular value truncation (Neufeld et al., 2012). In the nonlinear case (manifold learning), the loss functions deals with a matrix $\hat{\mathbf{D}} \in \mathbb{R}^{n\times n}$ that encodes the (dis)similarity between

1. Notations: Bold small letters $\mathbf{x}, \mathbf{y}$ are vectors. Bold capital letters $\mathbf{A}, \mathbf{B}$ are matrices. Calligraphic and double bold capital letters $\mathcal{X}$, $\mathcal{Y}$, $\mathbb{X}$, $\mathbb{Y}$ denote sets and/or spaces. Symmetric positive definite (SPD) and semi-definite (SPSD) matrices are denoted by $\mathbf{A} \succ 0$ and $\mathbf{A} \succeq 0$ respectively. $\mathrm{tr}(\cdot)$ is the matrix trace.
2. For the Mahalanobis distance, $\mathbf{A}$ is the inverse of the data's covariance matrix.

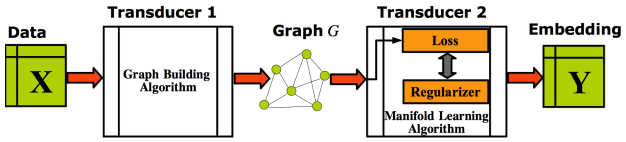


Figure 1: Standard manifold learning methods divide learning into two stages. First, Transducer 1 transforms the data matrix $\mathbf{X}$ into a graph G with weight matrix $\mathbf{W}$. Second, Transducer 2 defines a (dis)similarity matrix $\hat{\mathbf{D}}(\mathbf{W})$ and embeds graph G into a low dimensional representation $\mathbf{Y}$. Note that for most manifold learning algorithms, the input for Transducer 1 is in fact the metric space $(\mathbf{X}, \|\cdot\|_2)$. In this work, we replace $(\mathbf{X}, \|\cdot\|_2)$ with the pilot metric space $(\mathbb{X}, d_{\mathbb{X}})$ to mitigate the shortcomings of Euclidean geometry when dealing with real world data.

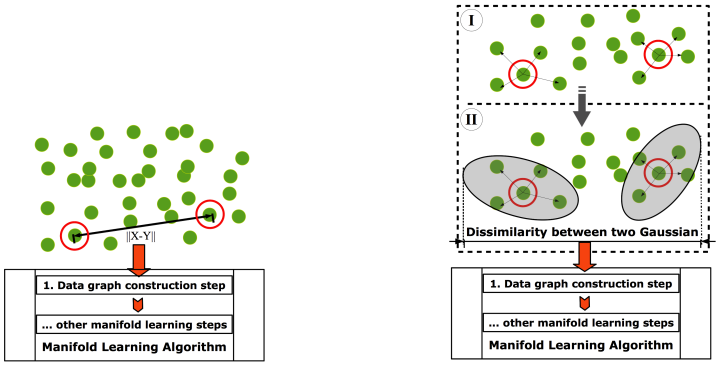
all pairs of points in a nonlinear manner. The matrix $\hat{\mathbf{D}}$, however, is induced by a weighted graph G through its adjacency and weight matrices. Most work on manifold learning algorithms has in fact focused on issues that are independent from the graph construction step; see Figure (1). Our observation is that any misleading information in the graph construction step, due to any of the aforementioned difficulties, will inevitably propagate to the loss function of the manifold learning algorithm.

In this paper we propose an algorithmic framework that overcomes these liabilities by learning a new input space for the manifold learner, such that the new input space, $\mathbb{X}$, characterizes the varying sample density in the original input space $\mathcal{X}$. In particular, we integrate the concept of local learning algorithms (Bottou and Vapnik, 1992), with parametric density estimation to learn from $\mathcal{D}$ a new metric space [3] $(\mathbb{X}, d_{\mathbb{X}})$ that becomes the (pilot) input space for the graph construction step in manifold learning. The proposed framework, depicted in Figure 2, redefines the proximity between two points in $\mathcal{D}$ based on the divergence between the local density surrounding each of the two points, then passes this proximity to the manifold learner. The set $\mathbb{X}$ contains all the parameters that define the local density for each point in $\mathcal{D}$, while the new proximity information is characterized by the divergence measure $d_{\mathbb{X}}$, which defines the metric space $(\mathbb{X}, d_{\mathbb{X}})$.

2. The Elements of the Set $\mathbb{X}$

We begin our discussion with a formal definition for the elements of $\mathbb{X}$. We assume that the input space $\mathcal{X}$ is *locally smooth*, hence it can be considered a smooth differentiable manifold that is locally Euclidean. Under this assumption, Euclidean geometry only holds in a small neighbourhood $\mathcal{N}$ around each point $\mathbf{x} \in \mathcal{X}$. In the finite sample setting, for each $\mathbf{x}_i \in \mathcal{D}$, the neighbourhood $\mathcal{N}(\mathbf{x}_i)$, or $\mathcal{N}_i$ for short, is the set of neighbouring points for $\mathbf{x}_i$, which can be defined using an $\epsilon-$ball or the m nearest neighbours of $\mathbf{x}_i$. Throughout the experiments in Section 6, we defined the $\mathcal{N}_i$'s using the m nearest neighbours for each point. This is

3. A metric space is an ordered pair $(\mathcal{X}, d)$ such that $\mathcal{X}$ is a non-empty abstract set (of any elements, whose nature is left unspecified), and d is a distance function, or a metric, defined as: $d : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, and the following axioms hold for all $a, b, c \in \mathcal{X}$: (i) $d(a, b) \geq 0$, (ii) $d(a, a) = 0$, (iii) $d(a, b) = 0$ iff $a = b$, (iv) Symmetry: $d(a, b) = d(b, a)$, and (v) The triangle inequality: $d(a, c) \leq d(a, b) + d(b, c)$. Semi-metrics satisfy axioms (i), (ii), and (iv) only. The axiomatic definition of metrics and semi-metrics, in particular axioms (i) and (ii), produce the positive semi-definiteness of d. Hence metrics and semi-metrics are positive semi-definite (PSD).



(a) (b)

Figure 2: (a) The standard distance used for data graph construction in manifold learning is Euclidean distance, indicated by the straight line between the points $\mathbf{x}$ and $\mathbf{y}$. (b) In the proposed framework (dashed black box), the distance between two points is defined in two steps: I. Each point defines a local neighbourhood by finding its m nearest neighbours. II. A Gaussian distribution with a low rank covariance is fit to each local neighbourhood. The distance between the two points is then the divergence between these two Gaussian distributions. This information is then conveyed to the graph construction step of the manifold learning algorithm.

similar to various learning algorithms that rely on local learning (Bottou and Vapnik, 1992; Ng et al., 2002; Vincent and Bengio, 2003; Belkin and Niyogi, 2003).

In principle, the neighbourhood size should grow slowly until it circumscribes the region where Euclidean geometry holds. Beyond this size, the local Euclidean assumption will break due to manifold curvature. Hence, if m is too small, the estimate for the local Euclidean subspace will be inaccurate, while if m is too large, the local linear structure will be smoothed out by the influence of distant points. In practice, m can be set either by using cross validation, grid search, or the method of (Brand, 2003).

For large data sets with very high dimensionality, finding the nearest neighbours for each $\mathbf{x}_i$ can be time consuming. In this case, approximate neighbours can be found using methods based on random projections. For example, locality-sensitive hashing (LSH) (Shakhnarovich et al., 2006) can be used where for h hash tables, each defined using k hash functions, the time complexity for finding m approximate nearest neighbours for all query points $\mathbf{q}_i$ is reduced to $O\left(nmh(k\tau + dn\epsilon)\right)$. Here τ is the time to evaluate the hash function on point $\mathbf{x}$, and $\epsilon \ll 1$ is the probability that the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$, $i \neq j$, is greater than a predefined threshold. Since τ is usually small, this complexity is substantially less than the $O(mn^2d^2)$ required for brute force search.

2.1. Local learning for manifold estimation

Local learning algorithms overcome a nonuniform data distribution by introducing a local adjustment mechanism that limits the influence to individual regions of the input space (Bottou and Vapnik, 1992). The Euclidean distance and the GQD do not have such a control mechanism, and hence do not take the varying sample density into consideration. We propose to introduce such a control mechanism to manifold learning in three steps:

1. Estimate the density for each $\mathcal{N}_i$, $i = 1, \ldots, n$. The parameters of these densities will define the elements of $\mathbb{X}$.

2. Define $d_{\mathbb{X}}$ as a dissimilarity measure on $\mathbb{X}$ (§ 4).

3. Use $(\mathbb{X}, d_{\mathbb{X}})$ to construct the neighbourhood graph for manifold learning.

Formally, let $\mathcal{N}_i \equiv \mathcal{N}(\mathbf{x}_i) = \{\mathbf{x}_1^i, \ldots, \mathbf{x}_m^i\}$, where $\mathbf{x}_j^i \in \mathcal{D}$, and $1 \leq i \leq n$. A reasonable density model for $\mathcal{N}_i$ under the local smoothness assumption is the Gaussian. That is for $\mathbf{x} \in \mathcal{N}(\mathbf{x}_i)$, then $\Pr(\mathbf{x}) = \mathcal{G}_i(\mathbf{x}; \mathbf{x}_i, \boldsymbol{\Sigma}_i)$, where $\mathcal{G}_i(\ \cdot\ ; \mathbf{x}_i, \boldsymbol{\Sigma}_i)$ is the Gaussian density centred at $\mathbf{x}_i$ for neighbourhood $\mathcal{N}_i$, $\boldsymbol{\Sigma}_i \in \mathbb{S}_{++}^{d\times d}$ is the sample covariance with respect to the mean $\mathbf{x}_i$, given by $\boldsymbol{\Sigma}_i = m^{-1} \sum_{\mathbf{x}\in\mathcal{N}_i} (\mathbf{x}-\mathbf{x}_i)(\mathbf{x}-\mathbf{x}_i)^\top$, and $\mathbb{S}_{++}^{d\times d}$ is the space of symmetric positive definite matrices. Note that such a local density model does not impose any constraints or assumptions on the global density for the data. Since in practice it might be that $d \gg m$, the sample covariance $\boldsymbol{\Sigma}_i$ will be a poor estimate for the true covariance, and generally rank deficient. Therefore, $\boldsymbol{\Sigma}_i$ can be replaced with a reliable low rank estimate $\hat{\boldsymbol{\Sigma}}_i \in \mathbb{S}_{++}^{d\times d}$ which shall be discussed in § 3.

In terms of local learning, $\boldsymbol{\mu}_i$ and $\hat{\boldsymbol{\Sigma}}_i$ are the local parameters that provide the means for coping with the uneven sample distribution. Ideally, each $\mathcal{G}_i$ defines a local neighbourhood around the point $\mathbf{x}_i$ with axes defined by the eigenvectors of $\boldsymbol{\Sigma}_i$, while its eigenvalues indicate the amount of data variance along each axis direction. If the data manifold is locally linear in the vicinity of $\mathbf{x}_i$, then all but the d_0 dominant eigenvalues will be very close to zero, while their associated leading eigenvectors will constitute the optimal variance preserving local coordinate system (Brand, 2003). In an ideal setting, a local maximum likelihood procedure will naturally capture this structure. However, to leverage the cases when $\boldsymbol{\Sigma}_i$ is degenerate for the above mentioned reasons, $\boldsymbol{\Sigma}_i$ is replaced with the low rank estimate $\hat{\boldsymbol{\Sigma}}_i$. This local Gaussian assumption at each point $\mathbf{x}_i$ is in the same spirit of stochastic neighbour embedding (SNE) (Hinton and Roweis, 2003), manifold charting (Brand, 2003), and manifold Parzen windows (Vincent and Bengio, 2003) for instance.

Given the set $\{\mathcal{N}_i\}_{i=1}^n$, their corresponding local densities $\mathscr{G} = \{\mathcal{G}_i\}_{i=1}^n$ characterize the varying sample density for $\mathcal{X}$ according to the parameters $\mathbf{x}_i$ and $\hat{\boldsymbol{\Sigma}}_i$. Since all local densities have the same parametric form, we define the set of 2-tuples $\{(\mathbf{x}_i, \hat{\boldsymbol{\Sigma}}_i)\}_{i=1}^n \subset \mathbb{X}$, where $\mathbb{X} \subset \mathbb{R}^d \times \mathbb{S}_{++}^{d\times d}$. Note that $\mathcal{N}(\mathbf{x})$ and $\hat{\boldsymbol{\Sigma}}$ are defined in an unsupervised manner. However, if auxiliary information is available in the form of labels or side information, the proposed approach can be extended to supervised and semi-supervised learning.

3. Low Rank Covariance Estimation

In various applications of machine learning such as computer vision and bioinformatics, one can be easily confronted with high dimensional data, thereby making d larger than the number of samples m in a neighbourhood $\mathcal{N}(\mathbf{x})$. The resulting scenario of *"large d small m"*, often, tends to break the standard assumptions of classical statistics, and can cause conventional estimators to behave poorly (Donoho, 2000). This problem is more serious when estimating a covariance matrix from a small number of samples. Indeed, accurate estimation of a covariance matrix from high dimensional data is a fundamental problem in statistics, since the number of parameters to be estimated grows quadratically with the

number of variables. In particular, for m samples and $d \gg m$, the sample covariance matrix will have a distorted eigen structure (Johnstone, 2001) and $d - m$ eigenvalues erroneously estimated as zeros (i.e. rank deficient). Learning a model under these conditions can be easily prone to overfitting, and yield poor generalization to out-of-sample points. In addition, computational tractability becomes another problem for handling such large matrices.[4]

Under the assumption that the data lies on a manifold of dimensionality $d_0 \ll d$, we would like to obtain a reliable estimate $\hat{\mathbf{\Sigma}}_i$ from the few m samples in $\mathcal{N}_i = \{\mathbf{x}_1^i, \ldots, \mathbf{x}_m^i\}$. In particular, we would like to ensure that $\hat{\mathbf{\Sigma}}_i$ has a rank at most $k \leq m \ll d$ by being decomposable into $\hat{\mathbf{\Sigma}}_i = \mathbf{B}_i \mathbf{B}_i^\top$ such that $\mathbf{B}_i \in \mathbb{R}^{d \times k}$. Besides being a reliable low rank estimate, such an estimate would also be computationally appealing (in space and time) since any vector–matrix product with $\hat{\mathbf{\Sigma}}_i$, as well as the space required to store it, is $O(dk)$.

Under the Gaussian modelling of $\mathcal{N}_i$, a maximum likelihood estimate of $\mathbf{\Sigma}_i$ will be:

$$\hat{\mathbf{\Sigma}}_i^* = \underset{\mathbf{\Sigma}_i \in \mathbb{S}_{++}^{d \times d}}{\arg\max}\ L(\mathcal{N}_i, \mathbf{\Sigma}_i), \text{ where} \tag{1}$$

$$L(\mathcal{N}_i, \mathbf{\Sigma}_i) = \prod_{\mathbf{x}_j \in \mathcal{N}_i}^{m} \Pr(\mathbf{x}_j) = \prod_{\mathbf{x}_j \in \mathcal{N}_i}^{m} \mathcal{G}(\mathbf{x}_j; \mathbf{x}_i, \mathbf{\Sigma}_i),$$

and $\mathcal{G}(\mathbf{x}_j; \mathbf{x}_i, \mathbf{\Sigma}_i) = (2\pi)^{-d/2} \det(\mathbf{\Sigma}_i)^{-1/2} \exp\{\frac{1}{2}(\mathbf{x}_j - \mathbf{x}_i)^\top \mathbf{\Sigma}_i^{-1} (\mathbf{x}_j - \mathbf{x}_i)\}$. Taking the logarithm of $L(\mathcal{N}_i, \mathbf{\Sigma}_i)$ and rearranging the terms, problem (1) will be:

$$\hat{\mathbf{\Sigma}}_i^* = \underset{\mathbf{\Sigma}_i \in \mathbb{S}_{++}^{d \times d}}{\arg\max}\ \mathcal{L}(\mathbf{\Sigma}_i), \text{ where} \tag{2}$$

$$\mathcal{L}(\mathbf{\Sigma}_i) = -\log\det(\mathbf{\Sigma}_i) - \operatorname{tr}(\mathbf{\Sigma}_i^{-1} \mathbf{C}_i), \tag{3}$$

and $\mathbf{C}_i$ is the sample covariance matrix of $\mathcal{N}_i$. Note that the mean vector for each $\mathcal{G}_i$ is the point of interest $\mathbf{x}_i$, and hence $\mathbf{C}_i = m^{-1} \sum_{\mathbf{X}_j \in \mathcal{N}_i} (\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)^\top$.

There are a few observations to make about that objective $\mathcal{L}(\mathbf{\Sigma}_i)$. First, $\mathcal{L}(\mathbf{\Sigma}_i)$ is not concave in $\mathbf{\Sigma}_i$, however with a change of variable $\mathbf{S}_i = \mathbf{\Sigma}_i^{-1}$, it becomes concave in $\mathbf{S}_i$. Second, for high dimensional data, $\mathbf{C}_i$ will be rank deficient and a poor estimate for the sample covariance. Hence, $\mathbf{C}_i$ can be replaced with the regularized estimate $\mathbf{R}_i$ which can be obtained by means of shrinkage estimators.[5] Taking these two remarks into consideration, and adding the constraint that $\operatorname{rank}(\mathbf{S}_i) \leq k$, we obtain the constrained optimization problem:[6]

$$\begin{aligned} \max \quad & \ell(\mathbf{S}_i), \quad \mathbf{S}_i \in \mathbb{S}_{++}^{d \times d} \\ s.t. \quad & \operatorname{rank}(\mathbf{S}_i) \leq k, \end{aligned} \tag{4}$$

where

$$\ell(\mathbf{S}_i) = \log\det(\mathbf{S}_i) - \operatorname{tr}(\mathbf{S}_i \mathbf{R}_i). \tag{5}$$

4. A simple trick when $d \gg n$ is to project the data on the first n principal components of the total scatter matrix $\mathbf{S}_t$. This results in zero loss of discriminatory information since the null space of $\mathbf{S}_t$ contains no discriminatory information.
5. $\mathbf{R}_i$ can be obtained by shrinking $\mathbf{C}_i$ towards a scaled identity matrix: $\mathbf{R}_i = (1-\gamma)\mathbf{C}_i + \gamma \operatorname{tr}(\mathbf{C}_i) d^{-1} \mathbf{I}$, or the diagonal entries of $\mathbf{C}_i$: $\mathbf{R} = (1-\gamma)\mathbf{C}_i + \gamma \operatorname{diag}(\mathbf{C}_i)$, where $\gamma \in (0, 1)$ is the intensity coefficient.
6. It is well known that the optimal solution to problem (4) without the rank constraint is the regularized sample covariance matrix $\mathbf{R}_i$.

Algorithm 1 `ApproxLowRankSDP`

Require: Concave function ℓ with curvature constant C_ℓ and target accuracy ε.
  Initialize $\mathbf{S}^{(1)} = \mathbf{v}_0\mathbf{v}_0^\top$ with an arbitrary unit length vector $\mathbf{v}_0$.
  for $t = 1 : \infty$ **do**
    Compute $\mathbf{v}_t = \texttt{ApproxEig}(\nabla\ell(\mathbf{S}^{(t)}), \frac{C_\ell}{t^2})$
    Set $\alpha_t = \min(1, \frac{2}{t})$
    Set $\mathbf{S}^{(t+1)} = \mathbf{S}^{(t)} + \alpha_t(\mathbf{v}_t\mathbf{v}_t^\top - \mathbf{S}^{(t)})$
  end for

Since $\mathbf{S}_i \in \mathbb{S}_{++}^{d\times d}$, problem (4) is equivalent to:

$$\begin{aligned} \max \quad & \ell(\mathbf{S}_i), \quad \mathbf{S}_i \in \mathbb{S}_{++}^{d\times d} \\ s.t. \quad & \operatorname{tr}(\mathbf{S}_i) = k. \end{aligned}$$

By rescaling all the entries in $\mathbf{S}_i$ by $1/k$ we obtain:

$$\begin{aligned} \max \quad & \ell(\mathbf{S}_i), \quad \mathbf{S}_i \in \mathcal{S} \\ s.t. \quad & \operatorname{tr}(\mathbf{S}_i) = 1, \end{aligned} \tag{6}$$

where $\mathcal{S} = \{\mathbf{S} \in \mathbb{S}_{++}^{d\times d} \mid \operatorname{tr}(\mathbf{S}) = 1\}$ is the set of unit trace symmetric PSD matrices. That is, the objective is to maximize the concave function $\ell(\mathbf{S}_i)$ over the bounded semidefinite cone $\mathcal{S}$. This can be solved efficiently, as explained in the following section.

3.1. Approximate solution to SDPs

Problem (6) is a special case of a semidefinite program (SDP) where the objective is to maximize the concave function $\ell(\mathbf{S}_i)$ over the bounded semidefinite cone $\mathcal{S}$. This particular type of program can be efficiently solved using the recent first order optimization scheme for SDPs (Hazan, 2008). In particular, Hazan's algorithm provides an ε-approximate[7] solution to the SDP problem with a strong approximation guarantee in the sense of obtaining ε-small primal–dual error after at most $O(1/\varepsilon)$ iterations. In this algorithm, each iteration only involves the calculation of a single approximate eigenvector of a matrix $\mathbf{A} \in \mathbb{S}_{++}^{d\times d}$. Thus, after t iterations the algorithm attains a $\frac{1}{t}$-approximate solution with rank at most t.

Hazan's algorithm, shown in Algorithm (1), is a first order gradient decent (a.k.a Frank–Wolfe) type algorithm that proceeds in iterations, such that in each iteration the solution is provably better than the preceding one, with rank increased by at most one. The resulting approximate solution has the favourable property of being low rank and decomposable into $\mathbf{B}\mathbf{B}^\top$ where $\mathbf{B} \in \mathbb{R}^{d\times k}$, and $k \ll d$. Furthermore, the algorithm is free of tuning parameters, easy to implement and parallelize as it only uses the power method (alternatively Lanczos steps) to approximate the largest eigenvector of a matrix. It is worth noting the following with regards to Algorithm (1). First, the curvature constant C_ℓ (or the modulus of convexity) determines the convergence speed of the algorithm and is defined as:

$$C_\ell = \sup_{\substack{x,z\in\mathcal{S},\ \alpha\in\mathbb{R} \\ y=x+\alpha(z-x)}} \tfrac{1}{\alpha^2}[\ell(x) - \ell(y) - \langle y - x, -\nabla\ell(x)\rangle].$$

7. An ε-approximate solution to an SDP over the bounded cone is a PSD matrix with trace equals one, which satisfies all the constraints up to an additive term ε.

Note that C_ℓ is upper bounded by the largest eigenvalue of the Hessian of $-\ell$, and for various types of functions, C_ℓ is usually small. Second, the function $\texttt{ApproxEig}(\mathbf{A}, \tau)$ is an approximate eigenvalue solver that computes the leading eigenvector of matrix $\mathbf{A}$ with the desired accuracy τ; i.e. it yields a unit length vector $\mathbf{v}$ such that $\mathbf{v}^\top \mathbf{A}\mathbf{v} \geq \lambda_{\max}(\mathbf{A}) - \tau$. Last, note that for $\ell(\mathbf{S}_i)$ in (5), the gradient $\nabla\ell(\mathbf{S}_i) = \mathbf{S}_i^{-1} - \mathbf{R}_i$ is also a SPSD matrix.

So far we have defined the space $\mathbb{X}$ which will be the input to the graph construction step of the manifold learning algorithm. To convey the information on the nonuniform density characterized in $\mathbb{X}$ to the graph algorithm, we need to define dissimilarity measures between the elements of $\mathbb{X}$. This is introduced in the following section.

4. The Divergence Measure $d_{\mathbb{X}}$

The measure $d_{\mathbb{X}}$ conveys the dissimilarity between two local densities, $\mathcal{G}_i$ and $\mathcal{G}_j$, which describe the local neighbourhoods $\mathcal{N}_i$ and $\mathcal{N}_j$ around points $\mathbf{x}_i$ and $\mathbf{x}_j$ respectively. Note that $d_{\mathbb{X}}$ measures the difference between two local coordinate systems located at $\mathbf{x}_i$ and $\mathbf{x}_j$ in terms of: (i) the location, specified by $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$, and (ii) the scaling and orientation, specified by the eigenvectors of $\hat{\boldsymbol{\Sigma}}_i$ and $\hat{\boldsymbol{\Sigma}}_j$, which define the axes of each local coordinate system. It is important, therefore, to understand the properties of $d_{\mathbb{X}}$, how these properties can affect the topology estimation, and how they affect the final embedding. Before proceeding to these questions, let us formally define the measure $d_{\mathbb{X}}$.

Since the elements of $\mathbb{X}$ are the parameters of the local Gaussians $\mathscr{G} = \{\mathcal{G}_i\}_{i=1}^n$, a natural measure for the dissimilarity between two densities is the divergence. For $\mathcal{G}_i$ and $\mathcal{G}_j$, some well known divergence measures with closed form expressions are: (1) The symmetric KL (or Jeffreys) divergence:

$$d_J(\mathcal{G}_i, \mathcal{G}_j) = \tfrac{1}{2}\mathbf{u}^\top \boldsymbol{\Psi}\mathbf{u} + \tfrac{1}{2}\mathrm{tr}\{\hat{\boldsymbol{\Sigma}}_i^{-1}\hat{\boldsymbol{\Sigma}}_j + \hat{\boldsymbol{\Sigma}}_j^{-1}\hat{\boldsymbol{\Sigma}}_i\} - d, \tag{7}$$

where $\boldsymbol{\Psi} = (\hat{\boldsymbol{\Sigma}}_i^{-1} + \hat{\boldsymbol{\Sigma}}_j^{-1})$, and $\mathbf{u} = (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$. (2) The Bhattacharyya distance d_B:

$$d_B(\mathcal{G}_i, \mathcal{G}_j) = \tfrac{1}{8}\mathbf{u}^\top \boldsymbol{\Gamma}^{-1}\mathbf{u} + \tfrac{1}{2}\ln\left[|\hat{\boldsymbol{\Sigma}}_i|^{-\frac{1}{2}}|\hat{\boldsymbol{\Sigma}}_j|^{-\frac{1}{2}}|\boldsymbol{\Gamma}|\right], \tag{8}$$

where $\boldsymbol{\Gamma} = (\frac{1}{2}\hat{\boldsymbol{\Sigma}}_i + \frac{1}{2}\hat{\boldsymbol{\Sigma}}_j)$. (3) The Hellinger distance $d_H = \sqrt{1 - \rho(\mathcal{G}_i, \mathcal{G}_j)}$, where ρ is the Bhattacharyya coefficient: $\rho(\mathcal{G}_i, \mathcal{G}_j) = |\boldsymbol{\Gamma}|^{-1/2}|\hat{\boldsymbol{\Sigma}}_i|^{1/4}|\hat{\boldsymbol{\Sigma}}_j|^{1/4}\exp\{-\frac{1}{8}\mathbf{u}^\top\boldsymbol{\Gamma}^{-1}\mathbf{u}\}$. Note that $d_B(\mathcal{G}_i, \mathcal{G}_j) = -\log[\rho(\mathcal{G}_i, \mathcal{G}_j)]$. (4) Recently Abou-Moustafa and Ferrie (2012) proposed variants of d_J and d_B where the second term in (7) and (8) is replaced with a metric, which yields the Jeffreys-Riemann and the Bhattacharyya-Riemann metrics:

$$d_{J\mathcal{R}}(\mathcal{G}_i, \mathcal{G}_j) = (\tfrac{1}{2}\mathbf{u}^\top \boldsymbol{\Psi}\mathbf{u})^{\frac{1}{2}} + d_{\mathcal{R}}(\hat{\boldsymbol{\Sigma}}_i, \hat{\boldsymbol{\Sigma}}_j), \tag{9}$$

$$d_{B\mathcal{R}}(\mathcal{G}_i, \mathcal{G}_j) = (\mathbf{u}^\top \boldsymbol{\Gamma}^{-1}\mathbf{u})^{\frac{1}{2}} + d_{\mathcal{R}}(\hat{\boldsymbol{\Sigma}}_i, \hat{\boldsymbol{\Sigma}}_j), \tag{10}$$

where $d_{\mathcal{R}}(\hat{\boldsymbol{\Sigma}}_i, \hat{\boldsymbol{\Sigma}}_j) = \mathrm{tr}\{\log^2 \boldsymbol{\Lambda}(\hat{\boldsymbol{\Sigma}}_i, \hat{\boldsymbol{\Sigma}})\}^{1/2}$ is the Riemannian metric for symmetric positive definite matrices (Rao, 1945) such that $\boldsymbol{\Lambda}(\hat{\boldsymbol{\Sigma}}_i, \hat{\boldsymbol{\Sigma}}_j) = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$ is the generalized eigenvalue matrix for the generalized eigenvalue problem $\hat{\boldsymbol{\Sigma}}_i\boldsymbol{\Phi} = \boldsymbol{\Lambda}\hat{\boldsymbol{\Sigma}}_j\boldsymbol{\Phi}$.

Note that $d_{\mathcal{R}}$ is the metric for the manifold $\mathbb{S}_{++}^{d\times d}$ and hence it satisfies all metric axioms. The measure $d_{\mathbb{X}}$, therefore, can be any of the above mentioned divergence measures and in the following, we shall consider their metric properties and how they affect the final embedding obtained by a manifold learning algorithm.

4.1. Manifold learning using $(\mathbb{X}, d_{\mathbb{X}})$

The divergence between any two probability distributions, P_1 and P_2, has the following properties: $Div(P_1, P_2) \geq 0$, and $Div(P_1, P_2) = 0$ iff $P_1 = P_2$ (Kullback, 1997). Therefore, by definition, Div satisfies axioms (i), (ii), and (iii) of metrics (see Footnote 3), and in general, is not symmetric nor does it satisfy the triangle inequality. However, for the particular cases of the above divergence measures, all are symmetric, and hence axiom (iv) is also satisfied. Unfortunately, the triangle inequality does not hold for the KL divergence, nor does it for the Bhattacharyya distance d_B (Kailath, 1967). For d_H, and the variants $d_{J\mathcal{R}}$ and $d_{B\mathcal{R}}$, they all satisfy the triangle inequality. The metric properties of divergence measures are intimately related to the positive semi-definiteness of the affinity matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ extracted from the graph's adjacency matrix. To see this, consider for instance the two algorithms; classical multidimensional scaling (cMDS) (Young and Householder, 1938), and Laplacian eigenmaps (LEM) (Belkin and Niyogi, 2003). Further, let $\mathbf{D} \in \mathbb{R}^{n \times n}$ be the matrix of pairwise divergences where $\mathbf{D}_{ij} = Div(\mathcal{G}_i, \mathcal{G}_j)$, $\forall i, j$, and Div is a symmetric PSD divergence measure.

For cMDS, the affinity matrix $\mathbf{A}$ is defined as $\mathbf{A}_{ij} = -\frac{1}{2}\mathbf{D}_{ij}^2$, $\forall i, j$. The matrix $\mathbf{A}$ is guaranteed to be PSD *if and only if* $Div(\mathcal{G}_i, \mathcal{G}_j)$ is a metric; in particular if it satisfies the triangle inequality. This result is due to Theorem (3) in (Young and Householder, 1938). Therefore, Div in the case of cMDS can be d_H, $d_{J\mathcal{R}}$, or $d_{B\mathcal{R}}$ since they all metrics. For LEM, and for input vectors $\mathbf{x}_i, \mathbf{x}_j$, the affinity matrix $\mathbf{A}$ is defined as $\mathbf{A}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, $\forall i, j$, where K is a symmetric PSD kernel that measures the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. From Mercer kernels, it is known that $\mathbf{A}$ is PSD *if and only if* K is symmetric and PSD. A possible kernel for $\mathcal{G}_i$ and $\mathcal{G}_j$ using a symmetric Div is: $K(\mathcal{G}_i, \mathcal{G}_j) = \exp\{-\frac{1}{\sigma} Div(\mathcal{G}_i, \mathcal{G}_j)\}$, where $\sigma > 0$ is a parameter that scales the affinity between two densities. Since Div is PSD and symmetric, then $K(\mathcal{G}_i, \mathcal{G}_j)$ is PSD and symmetric as well. This simple fact is due to Theorems (2) and (4) in (Schoenberg, 1938) and a discussion on these particular kernels in the context of LEM can be found in (Abou-Moustafa et al., 2011). Therefore, for LEM, a symmetric PSD affinity matrix can be defined as $\mathbf{A}_{ij} = K(\mathcal{G}_i, \mathcal{G}_j)$, $\forall i, j$, and using any symmetric Div to define the kernel K.

4.2. Neighbourhood estimation using $(\mathbb{X}, d_{\mathbb{X}})$

We will make a slight abuse of the notation and let: $d_{\mathbb{X}} \equiv d_{\mathbb{X}}(\mathbf{x}_i, \mathbf{x}_j) = Div(\mathcal{G}_i, \mathcal{G}_j)$, to imply that querying the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ with respect to the space $\mathbb{X}$ returns the divergence between their respective local densities, where Div is d_J, d_B, d_H, $d_{J\mathcal{R}}$, or $d_{B\mathcal{R}}$. The expressions for Div in Equations (7), (8), (9), and (10) are summations of two terms; the first term is for the difference in means $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ weighted by a symmetric positive definite matrix, and the second term is for the discrepancy between $\hat{\boldsymbol{\Sigma}}_i$ and $\hat{\boldsymbol{\Sigma}}_j$ (Kullback, 1997). Now consider how $d_{\mathbb{X}}$ behaves in different scenarios and how it will affect the neighbourhood graph construction for manifold learning. If $\boldsymbol{\mu}_i = \boldsymbol{\mu}_j = \boldsymbol{\mu}$ (or $\boldsymbol{\mu}_i \approx \boldsymbol{\mu}_j$), then the first term in (7), (8), (9), and (10) will be zero (or very small), and $d_{\mathbb{X}}(\mathbf{x}_i, \mathbf{x}_j)$ will be mainly determined by the dissimilarity in the covariances. If $\hat{\boldsymbol{\Sigma}}_i = \hat{\boldsymbol{\Sigma}}_j = \boldsymbol{\Sigma}$ (or $\hat{\boldsymbol{\Sigma}}_i \approx \hat{\boldsymbol{\Sigma}}_j$), then the second term in (7), (8), (9), and (10) will be zero (or very small) and $d_{\mathbb{X}}(\mathbf{x}_i, \mathbf{x}_j)$ reduces to the Mahalanobis distance. Further, if $\hat{\boldsymbol{\Sigma}}_i = \hat{\boldsymbol{\Sigma}}_j = \mathbf{I}$, then $d_{\mathbb{X}}(\mathbf{x}_i, \mathbf{x}_j)$ reduces to the Euclidean distance between $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$.

How does $d_{\mathbb{X}}$ affect the neighbourhood estimation? Any two points $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathcal{D}$ (equivalently two nodes on the graph) will be close to each other, *if and only if* $\boldsymbol{\mu}_i \approx \boldsymbol{\mu}_j$ and $\hat{\boldsymbol{\Sigma}}_i \approx \hat{\boldsymbol{\Sigma}}_j$. That is, it is not sufficient that $\|\mathbf{x}_i - \mathbf{x}_j\|_2$ is small. This new meaning for the distance between points is more restrictive and different from the Euclidean distance and the GQD, which are special cases from $d_{\mathbb{X}}(\mathbf{x}_i, \mathbf{x}_j)$. Note that $d_{\mathbb{X}}(\mathbf{x}_i, \mathbf{x}_j)$ has an effect only on $\mathbf{x}_i$ and $\mathbf{x}_j$, but not on any other points in $\mathcal{D}$. This is due to the nature of local learning employed to learn $(\mathbb{X}, d_{\mathbb{X}})$, together with the nature of $d_{\mathbb{X}}$ as a divergence measure. Note also that our previous discussion did not rely on labels nor side information, and the whole process was unsupervised. The only assumption made was the smoothness of $\mathcal{X}$, and that it is locally Euclidean to bootstrap $(\mathbb{X}, d_{\mathbb{X}})$.

Selecting parameters m and γ: The metric space $(\mathbb{X}, d_{\mathbb{X}})$ is adaptive and mainly controlled by m which is data dependent and task dependent. The intensity parameter γ is less influential than m since it only occurs in $\mathbf{R}_i$ in Equation (5) to avoid numerical problems in Algorithm (1) (Hazan, 2008). Hence γ should be small and sufficient to ensure that $\mathbf{R}_i$ is PSD. This is unlike selecting m which controls the resolution of $\mathbb{X}$. In practice, however, learning algorithms have a clear objective function to optimize, and m should be chosen in accordance to this objective function. For instance, in classification, m can be selected to minimize the classification error on a training or a validation set. In clustering, assuming k clusters and using LEM (see § 6), the best value for m is the one that maximizes the sum of the first k eigenvalues (i.e. the energy) of LEM's affinity matrix[8] (Luxburg, 2007).

Computational complexity: It is important to note that the computational complexity for $(\mathbb{X}, d_{\mathbb{X}})$ is independent from that of the manifold learning algorithm. Computing $\mathbb{X}$ using LSH (see § 2) is $O\left(nmh(k\tau + dn\epsilon)\right)$, while the low rank estimate for all n points using Hazan's algorithm is $O(ndk)$, and computing $d_{\mathbb{X}}$ is $O(n(n-1)dk/2)$, where $k \ll d$.

5. Related Work

Other work has considered the problem of inferring good neighbourhood graph structure for manifold learning. For example, the well known ISOMAP method extends cMDS by redefining the distance metric between any two points as the shortest path connecting them on the neighbourhood graph (Tenenbaum et al., 2000). The neighbourhood graph approximates the underlying manifold topology, while the shortest path distance approximates the geodesic distance between the points. By passing this distance to cMDS, the algorithm then finds an Euclidean embedding that preserves this geodesic distance between points. Note that the weights on the neighbourhood graph in ISOMAP are Euclidean distances. Since the shortest path distance satisfies the triangle inequality (Cormen et al., 2001) (Lemma 24.10), ISOMAP's inferred geodesic distances must preserve the triangle inequality in the embedding space.[9] Note that the same argument follows when $(\mathbb{X}, d_{\mathbb{X}})$ is combined with ISOMAP and $d_{\mathbb{X}}$ is a metric. In § 7, we will discuss why this can be an interesting property.

ISOLLE (Varini et al., 2006) extends LLE (Roweis and Saul, 2000) by finding the nearest neighbours for each point using the geodesic distance defined by ISOMAP. Unfortunately, in the finite sample setting, the uneven sample distribution and noise in the data can greatly

8. For LEM, in a perfect k clusters setting, the first k eigenvalues of the affinity matrix are all equal to 1.
9. See the discussion on metric properties in § 4.1.

Table 1: Attributes of the twelve data sets used in our experiments.

Data set	size	features	classes	Data set	size	features	classes
Balance	625	4	3	Lymphography	148	18	4
Bupa	345	6	2	NewThyroid	215	5	3
Glass	214	9	6	Wine	178	13	3
Ionosphere	351	33	2	Corel	500	36	5
Segment	2310	18	7	Sonar	208	60	2

impact the topological stability of ISOMAP and all other manifold learning methods (Balasubramanian et al., 2002). This is our rationale for proposing $(\mathbb{X}, d_{\mathbb{X}})$ as an adaptive mechanism that explicitly captures the varying sample density in the data, and leverages the impact of noise. Carreira-Perpinan and Zemel (2004) considers graph ensembles instead of a single graph as an estimate for the manifold topology, where the ensemble is obtained from various perturbed versions of the data. The main limitation of this approach is its high computational complexity (in space and time) when considering large data sets. Recently, Gashler and Martinez (2011) proposed the CycleCut algorithm to remove shortcut connections in order to enhance the manifold estimation. Note that this approach patches some of the problems caused by the Euclidean distance, while our approach redefines a new input space for manifold learning that has a different geometry and distance between points.

Another related approach is the Graph–DBD algorithm (Bijral et al., 2011) for computing density based distances (DBDs). Here, a distance between two points on a manifold is defined as the path connecting these two points such that, paths passing through high density regions should be much shorter than paths passing through low density regions. However, since there are possibly various paths connecting any two points, the DBD is defined as the shortest path connecting these two points. Intuitively, if the distance between a point and its few nearest neighbours is small, then it is expected that the neighbourhood is a high density region. Algorithmically, a weight matrix $\mathbf{W}$ is computed for all points such that $\mathbf{W}_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_p^q$, where $q > 1$. The matrix $\mathbf{W}$ is the adjacency matrix of a fully connected graph with points $\{\mathbf{x}_i\}_{i=1}^n$ as its vertices. Following this definition, the DBD between $\mathbf{x}_i$ and $\mathbf{x}_j$ is the shortest path between nodes i and j on the graph defined by $\mathbf{W}$.

Similar to Graph–DBD, our proposed metric space $(\mathbb{X}, d_{\mathbb{X}})$ takes into considerations regions of high and low density when computing the distance between any two points. In fact, $(\mathbb{X}, d_{\mathbb{X}})$ explicitly characterizes the local density for each point through the local low rank covariance estimate $\hat{\mathbf{\Sigma}}_i$. Recalling § 4.2, note that the second term in Equations (7) – (10), acts as a penalty when the local densities are not similar, thereby increasing the final distance between the points. If two points are very close each to other and have very similar local densities, then the second term in Equations (7) – (10) is zero, and the final distance is very small. Note that parameter q in Graph–DBD controls the resolution of the manifold structure sought, which is similar to the neighbourhood size m in our approach.

6. Experiments

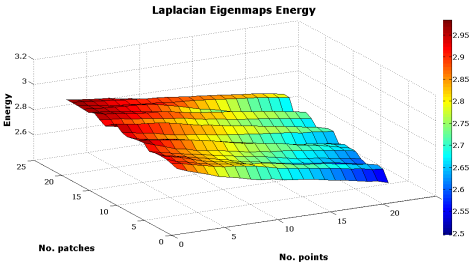
We performed a series of extensive experiments to test the validity and efficacy of the input space $(\mathbb{X}, d_{\mathbb{X}})$ for four different manifold learning methods; ISOMAP (ISM) (Tenenbaum et al., 2000), LEM (Belkin and Niyogi, 2003), LLE (Roweis and Saul, 2000), and tSNE

Table 2: k–Means clustering accuracy (ARI, NMI and PUR) for all algorithms described in the text on the twelve UCI data sets. Best performers are marked in bold.

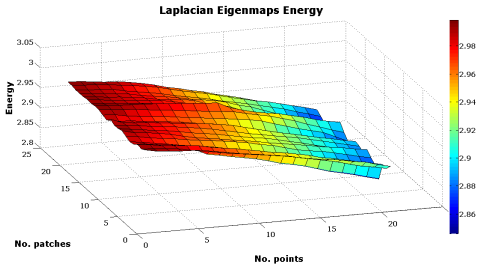
	ARI				NMI				PUR			
Data set	**Euc**	**PCA**	**ISM**	**𝕏–ISM**	**Euc**	**PCA**	**ISM**	**𝕏–ISM**	**Euc**	**PCA**	**ISM**	**𝕏–ISM**
balance	0.161	0.178	0.226	**0.387**	0.137	0.125	0.183	**0.273**	0.659	0.696	0.701	**0.763**
bupa	0.005	0.005	0.007	**0.014**	0.001	0.001	0.001	**0.017**	0.58	0.58	0.58	0.58
glass	0.255	0.246	0.251	**0.435**	0.378	0.357	0.362	**0.428**	0.589	0.547	0.626	**0.673**
iono.	0.178	0.168	0.149	**0.182**	0.131	0.124	0.104	**0.129**	0.712	0.707	0.695	**0.715**
iris	0.73	0.716	0.759	**0.818**	0.751	0.736	0.796	**0.804**	0.893	0.887	0.907	**0.933**
lymph.	0.123	0.116	0.238	**0.275**	0.142	0.128	0.181	**0.205**	0.703	0.703	0.75	**0.784**
newthy.	0.579	0.568	0.579	**0.647**	0.485	0.48	0.485	**0.497**	0.86	0.856	0.86	**0.884**
wdbc	0.491	0.491	0.497	**0.707**	0.422	0.422	0.427	**0.582**	0.854	0.854	0.856	**0.921**
wine	0.371	0.371	0.363	**0.421**	0.429	0.429	0.432	**0.48**	0.702	0.702	0.713	**0.747**
corel	0.199	0.163	0.176	**0.237**	0.288	0.242	0.263	**0.321**	0.488	0.462	0.49	**0.562**
segment	0.339	0.218	0.273	**0.592**	0.463	0.372	0.387	**0.666**	0.508	0.459	0.491	**0.735**
sonar	0.006	0.006	0.004	**0.068**	0.009	0.009	0.027	**0.05**	0.553	0.553	0.534	**0.635**
Data set	**Euc**	**PCA**	**LEM**	**𝕏–LEM**	**Euc**	**PCA**	**LEM**	**𝕏–LEM**	**Euc**	**PCA**	**LEM**	**𝕏–LEM**
balance	0.161	0.178	0.034	**0.416**	0.137	0.125	0.034	**0.289**	0.659	0.696	0.467	**0.778**
bupa	0.005	0.005	0.012	**0.011**	0.001	0.001	0.008	**0.016**	0.58	0.58	0.58	0.58
glass	0.255	0.246	0.232	**0.262**	0.378	0.357	0.353	**0.383**	**0.589**	0.547	0.533	0.565
iono.	0.178	0.168	0.11	**0.165**	0.131	0.124	0.068	**0.151**	0.712	0.707	0.67	**0.729**
iris	0.73	0.716	0.759	**0.904**	0.751	0.736	0.796	**0.88**	0.893	0.887	0.907	**0.967**
lymph.	0.123	0.116	0.128	**0.237**	0.142	0.128	0.138	**0.191**	0.703	0.703	0.709	**0.757**
newthy.	0.579	0.568	0.366	**0.674**	0.485	0.48	0.303	**0.501**	0.86	0.856	0.8	**0.884**
wdbc	0.491	0.491	0.047	**0.725**	0.422	0.422	0.049	**0.604**	0.854	0.854	0.659	**0.926**
wine	0.371	0.371	0.359	**0.432**	0.429	0.429	0.419	**0.445**	0.702	0.702	0.713	**0.736**
corel	0.199	0.163	0.204	**0.259**	0.288	0.242	0.288	**0.328**	0.488	0.462	0.51	**0.586**
segment	0.339	0.218	0.577	**0.581**	0.463	0.372	0.666	**0.689**	0.508	0.459	0.724	**0.759**
sonar	0.006	0.006	0.004	**0.053**	0.009	0.009	0.027	**0.075**	0.553	0.553	0.534	**0.62**
Data set	**Euc**	**PCA**	**LLE**	**𝕏–LLE**	**Euc**	**PCA**	**LLE**	**𝕏–LLE**	**Euc**	**PCA**	**LLE**	**𝕏–LLE**
balance	0.161	0.178	0.344	**0.397**	0.137	0.125	0.297	**0.333**	0.659	0.696	0.749	**0.814**
bupa	0.005	0.005	0.105	**0.146**	0.001	0.001	0.084	**0.105**	0.58	0.58	0.664	**0.693**
glass	0.255	0.246	0.152	**0.34**	0.378	0.357	0.179	**0.407**	0.589	0.547	0.519	**0.668**
iono.	0.178	0.168	0.065	**0.608**	0.131	0.124	0.03	**0.48**	0.712	0.707	0.641	**0.892**
iris	0.73	0.716	0.316	**0.922**	0.751	0.736	0.442	**0.901**	0.893	0.887	0.713	**0.973**
lymph.	0.123	0.116	0.144	**0.308**	0.142	0.128	0.172	**0.262**	0.703	0.703	0.723	**0.784**
newthy.	0.579	0.568	0.528	**0.781**	0.485	0.48	0.355	**0.669**	0.86	0.856	0.805	**0.935**
wdbc	0.491	0.491	0.067	**0.705**	0.422	0.422	0.038	**0.603**	0.854	0.854	0.666	**0.921**
wine	0.371	0.371	0.062	**0.492**	0.429	0.429	0.121	**0.486**	0.702	0.702	0.466	**0.792**
corel	0.199	0.163	0.041	**0.348**	0.288	0.242	0.08	**0.417**	0.488	0.462	0.332	**0.612**
segment	0.339	0.218	0.144	**0.601**	0.463	0.372	0.222	**0.672**	0.508	0.459	0.403	**0.774**
sonar	0.006	0.006	0.019	**0.227**	0.009	0.009	0.016	**0.173**	0.553	0.553	0.577	**0.74**
Data set	**Euc**	**PCA**	**tSNE**	**𝕏–tSNE**	**Euc**	**PCA**	**tSNE**	**𝕏–tSNE**	**Euc**	**PCA**	**tSNE**	**𝕏–tSNE**
balance	0.161	0.178	0.275	**0.34**	0.137	0.125	0.206	**0.232**	0.659	0.696	0.722	**0.742**
bupa	0.005	0.005	0.014	**0.052**	0.001	0.001	0.016	**0.041**	0.58	0.58	0.58	**0.617**
glass	0.255	0.246	0.21	**0.223**	0.378	0.357	0.346	**0.361**	0.589	0.547	0.607	**0.631**
iono.	0.178	0.168	0.124	**0.523**	0.131	0.124	0.098	**0.395**	0.712	0.707	0.678	**0.863**
iris	0.73	0.716	0.773	**0.941**	0.751	0.736	0.805	**0.93**	0.893	0.887	0.913	**0.98**
lymph.	0.123	0.116	0.142	**0.278**	0.142	0.128	0.152	**0.214**	0.703	0.703	0.709	**0.743**
newthy.	0.579	0.568	0.107	**0.625**	**0.485**	0.48	0.156	0.463	0.86	0.856	0.698	**0.865**
wdbc	0.491	0.491	0.415	**0.719**	0.422	0.422	0.405	**0.595**	0.854	0.854	0.822	**0.924**
wine	0.371	0.371	0.401	**0.452**	0.429	0.429	0.393	**0.453**	0.702	0.702	0.725	**0.742**
corel	0.199	0.163	0.241	**0.294**	0.288	0.242	0.318	**0.354**	0.488	0.462	0.536	**0.584**
segment	0.339	0.218	0.63	**0.637**	0.463	0.372	0.684	**0.693**	0.508	0.459	0.781	**0.784**
sonar	0.006	0.006	0.011	**0.079**	0.009	0.009	0.012	**0.065**	0.553	0.553	0.562	**0.644**

(van der Maaten and Hinton, 2008). Following previous work on spectral dimensionality reduction and clustering (Ng et al., 2002; Ding and Li, 2007), we assess the benefit of $(\mathbb{X}, d_{\mathbb{X}})$ for manifold learning by the accuracy of k-Means clustering. More specifically, we quantify the impact of the proposed input space by comparing clustering accuracy for the data in the embedding space obtained *before* and *after* using $(\mathbb{X}, d_{\mathbb{X}})$. Our hypothesis is that, for a manifold learning algorithm, say LLE, the clustering accuracy in the embedding space obtained by $(\mathbb{X}, d_{\mathbb{X}})$+LLE and denoted by $\mathbb{X}$–LLE, will always be higher than the accuracy in the embedding space obtained by LLE alone.

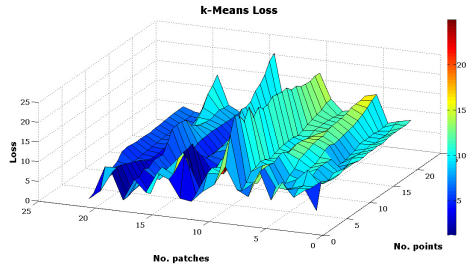
For the purpose of these experiments, we used twelve data sets, shown in Table (1), from the UCI Machine Learning Repository (Newman et al., 1998). We used k–Means for



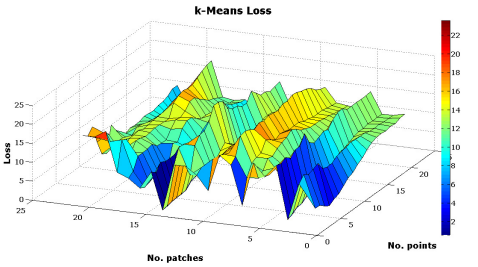
(a) LEM's energy for iris



(b) LEM's energy for wine



(c) k–Means loss for iris



(d) k–Means loss for wine

Figure 3: LEM's energy (a and b) and k–Means loss (c and d) for two real data sets, iris and wine, as functions of m (No. Points) and the neighbourhood size for LEM (No. Patches).

clustering the data in each of the embedding spaces mentioned above, and the number of clusters was assumed to be known. To leverage the impact of local minima for k–Means, the algorithm was restarted with 50 different initializations and the best clustering result was the one that minimized the sum of squared distances between the points and their respective cluster means. After that, the accuracy of the best clustering result was assessed using the adjusted rank index (ARI), the normalized mutual information index (NMI), and the purity index (PUR).

The baseline performance for these experiments was that of the k–Means algorithm in the Euclidean input space (Euc), and the subspace obtained by principal component analysis (PCA). The hyperparameters for each algorithm were tuned as follows. For PCA, the number of retained components constituted 98% of the total variance in the data. For ISOMAP, LLE, and LEM, the neighbourhood size was allowed to vary from 3 to 25 nearest neighbours. For LEM, and using a Gaussian kernel, there is an additional hyperparameter to optimize: σ the kernel width. Three values were considered for σ; from all the pairwise similarities in the affinity matrix, we selected the median, the 0.25 and the 0.75 of the distribution's quantile. The dimensionality d_0 for LEM was fixed to the number of classes in the data (Luxburg, 2007), and for ISOMAP and LLE it varied from 2 to $\lfloor d/2 \rfloor$ For tSNE, $d_0 = \{2, 3\}$, and the perplexity parameter varied from 10 to 35, with increments of 5 points.

For $(\mathbb{X}, d_{\mathbb{X}})$, there are two hyperparameters: m and γ. In all our experiments we used a regularized sample covariance of the form: $\mathbf{R}_i = \mathbf{\Sigma}_i + \gamma \mathbf{I}$, $\gamma = 1.0e-4$, while m varied from 3 to 25 neighbours. These values were found to work well in practice for all our experiments.

Table (2) shows the clustering accuracy (ARI, NMI and PUR) for all algorithms on the twelve UCI data sets. It can be seen that $(\mathbb{X}, d_{\mathbb{X}})$ yields a consistent improvement in the

clustering accuracy in the embedding spaces obtained by the four different algorithms. To assess the significance of the improvement caused by $(\mathbb{X}, d_{\mathbb{X}})$, we carried two significance tests on the results in Table (2), the nonparametric Friedman test, followed by a post Friedman multiple comparisons test, the Bonferroni–Dunn test (Demšar, 2006). These tests showed that $(\mathbb{X}, d_{\mathbb{X}})$ yielded significant improvements for ISOMAP (ARI, PUR), LEM (ARI, NMI, PUR), LLE (ARI, NMI, PUR), and tSNE (ARI, NMI, PUR). However, $\mathbb{X}$–ISM was not significantly different from ISM under the NMI index. These results demonstrate that $(\mathbb{X}, d_{\mathbb{X}})$ helps the manifold learner to better characterize regions of high density in the data revealing by that its latent class structure.

The results in Table (2) are not sensitive to γ for the reasons discussed in § 4.2. However, m has a strong influence on these results. Figure (3) depicts the influence of m (No. Points), and the neighbourhood size for LEM (No. Patches), on the energy for $\mathbb{X}$–LEM and the loss of k–Means in each embedding space obtained by $\mathbb{X}$–LEM. Since k is assumed to be known, LEM's energy is the sum of the first k eigenvalues of its affinity matrix, and the k–Means loss is the sum of squared distances between points and their respective cluster mean. Note that regions with high energy (dark red) for LEM nicely overlap with regions with minimal loss for k–Means (dark blue). This implies that in these regions $\mathbb{X}$–LEM reveals well defined compact clusters. Note also that, as expected from our assumptions, small values of m – i.e. small neighbourhoods where Euclidean geometry holds – yield better defined clusters.

It is worth noting that we have not discussed how to choose which divergence measure to use for a particular data set with a particular manifold learning algorithm. In the scope of this paper, however, we considered how metric properties of divergence measures impact the positive semi–definiteness of the affinity matrix for manifold learning in general (§ 4.1).

7. Discussion

We have proposed a framework for manifold learning algorithms that allows the liabilities of Euclidean geometry to be overcome when dealing with real world data. Our framework combines the concepts of local learning with parametric density estimation to learn, in an unsupervised manner, the metric space $(\mathbb{X}, d_{\mathbb{X}})$ which becomes a pilot input space to the graph construction step of the manifold learning process.

An interesting property of the proposed framework is that it is not only limited to the four manifold learners chosen here; in fact, it can be combined with any manifold learning algorithm that relies on a data graph to learn a low dimensional embedding for the data (Weinberger and Saul, 2004; Shaw and Jebara, 2009). Further, the framework does not intervene with, nor prevent the use of the Nyström method, which is a crucial technique for generalizing learned manifold structures to out of sample points, and scaling manifold learning methods. Another interesting property is that the framework can preserve the triangle inequality in the embedding space if $d_{\mathbb{X}}$ is a metric and the manifold learner is either LEM, cMDS or ISOMAP. Therefore, in the context of clustering on large data sets, the practitioner can make use of the advances in accelerating the $k-$Means algorithm based on the triangle inequality property (Elkan, 2003).

Last, since $(\mathbb{X}, d_{\mathbb{X}})$ is an adaptive metric space with tuning parameter m, it is possible to compute a distance matrix $\mathbf{D}_{\mathbb{X}} \in \mathbb{R}^{n \times n}$ using the divergence $d_{\mathbb{X}}$ – assuming a fully connected graph. In the same spirit of Graph–DBD (Bijral et al., 2011), $\mathbf{D}_{\mathbb{X}}$ can be used

with supervised or semi-supervised distance based methods such as nearest neighbours methods, Parzen windows methods, or kernel methods with distance based kernels (e.g. Gaussian kernel). Further, it is legitimate to ask whether it is possible to replace the weighted distance $\|\mathbf{x}_i - \mathbf{x}_j\|_p^q$ in the Graph–DBD algorithm with the divergence measure $d_{\mathbb{X}}(\mathbf{x}_i, \mathbf{x}_j)$ and proceed by computing all pairs shortest paths for all points; i.e. Graph–DBD based on $d_{\mathbb{X}}$. While this is an interesting research direction, we leave it as future work.

References

K. Abou-Moustafa and F. Ferrie. A note on metric properties of some divergence measures: The Gaussian case. *Proc. of the 4th ACML, JMLR W&CP*, 25:1–15, 2012.

K. Abou-Moustafa, M. Shah, F. De La Torre, and F. Ferrie. Relaxed exponential kernels for unsupervised learning. In *LNCS 6835, Pattern Recognition, Proc. of the 33rd DAGM Symp.*, pages 335–344. Springer, 2011.

M. Balasubramanian, E. Schwartz, J. Tenenbaum, V. de Silva, and J. Langford. The Isomap algorithm and topological stability. *Science*, 295(5552):7, 2002.

M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for data representation. *Neural Computation*, 15:1373–1396, 2003.

Y. Bengio, O. Delalleau, N. Le Roux, J-F. Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Computation*, 16:2197–2219, 2004.

A. S. Bijral, N. D. Ratliff, and N. Srebro. Semi-supervised learning with density based distances. In *Proc. of the 27th Conf. on UAI*, 2011.

L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, 1992.

M. Brand. Charting a manifold. In *Advances in NIPS 15*, 2003.

M. Carreira-Perpinan and R. Zemel. Proximity graphs for clustering and manifold learning. In *Advances in NIPS 18*, 2004.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, 2001.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *JMLR*, 7:1–30, 2006.

C. Ding and T. Li. Adaptive dimension reduction using discriminant analysis and K–means clustering. In *ACM Proc. of the 24th ICML*, 2007.

D. L. Donoho. High–dimensional data analysis: the curses and blessings of dimensionality. In *American Mathematical Society Conf. Math Challenges of the 21st Century*. 2000.

C. Elkan. Using the triangle inequality to accelerate k–means. In *ACM Proc. of the 20th ICML*, 2003.

M. Gashler and T. Martinez. Robust manifold learning with cyclecut. In *ACM Proc. of the 28th ICML*, 2011.

E. Hazan. Sparse approximate solutions to semidefinite programs. In *Proc. of the 8th Latin American Conf. on Theoretical Informatics (LATIN)*. Springer-Verlag, 2008.

G. Hinton and S. Roweis. Stochastic neighbor embedding. In *Advances in NIPS 15*, 2003.

I. M. Johnstone. On the distribution of the largest eigenvalue in principal components analysis. *The Annals of Statistics*, 29(2):pp. 295–327, 2001.

T. Kailath. The divergence and Bhattacharyya distance measures in signal selection. *IEEE Trans. on Communication Technology*, 15(1):52–60, 1967.

S. Kullback. *Information Theory and Statistics – Dover Edition*. Dover, New York, 1997.

U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

J. Neufeld, Y. Yu, X. Zhang, R. Kiros, and D. Schuurmans. Regularizers versus losses for nonlinear dimensionality reduction: A factored view with new convex relaxations. In *ACM Proc. of the 29th ICML*, 2012.

D. Newman, S. Hettich, C. Blake, and C. Merz. UCI Repository of Machine Learning Databases, 1998. `www.ics.uci.edu/∼mlearn/MLRepository.html`.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in NIPS 14*, 2002.

C. R. Rao. Information and the accuracy attainable in the estimation of statistical parameters. *Bull. Calcutta Math. Soc.*, (58):326–337, 1945.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding (LLE). *Science*, 290 (5500):2323–2326, 2000.

I. Schoenberg. Metric spaces and positive definite functions. *Trans. of the American Mathematical Society*, 44(3):522–536, 1938.

B. Scholkopf, A. Smola, and K-R Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. The MIT Press, Cambridge MA, 2006.

B. Shaw and T. Jebara. Structure preserving embedding. In *ACM Proc. of the 26th ICML*, 2009.

J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, November 2000.

L. van der Maaten and G. Hinton. Visualizing data using t–SNE. *JMLR*, 9:2579–2605, Nov. 2008.

C. Varini, A. Degenhard, and T. W. Nattkemper. ISOLLE: LLE with geodesic distance. *Neurocomputing*, 69(13–15):1768–1771, August 2006.

P. Vincent and Y. Bengio. Manifold Parzen windows. In *Advances in NIPS 15*, 2003.

K. Weinberger and L. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *IEEE Proc. of CVPR*, pages 988–995, 2004.

K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbour classification. In *Advances in NIPS 18*, 2006.

C. K.I. Williams. On a connection between kernel PCA and metric multidimensional scaling. *Machine Learning*, 46:11 – 19, 2002.

E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning with application to clustering with side–information. In *Advances in NIPS 15*, 2002.

G. Young and A. Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3(1):19–22, 1938.