DD2434 Project Group 94

Reimplementation of GPLVM

Anirudh Seth (aniseth@kth.se) Magnus Pierrau (mpierrau@kth.se)
Matej Buljan (buljan@kth.se) Prashant Maheshwari (pramah@kth.se)

January 2020

Abstract

This project builds on two research papers published in 2004 and 2005 by Neil D. Lawrence that present a novel dimensionality reduction (DR) technique called Gaussian Process Latent Variable Model (GPLVM) and compare it to other DR techniques known at the time. Here we reproduce the original results, confirming the claims made by Lawrence by using our own implementation of the methods being used and compare the results to more recent approaches to dimensionality reduction. Furthermore, we critically discuss how our results compare to those from the original papers and suggest improvements to the experimental setup..

1 Introduction

In his 2004 and 2005 papers, [1] and [2], Neil Lawrence presents a new method for dimensionality reduction (DR) that builds on the Probabilistic PCA (PPCA) model using Gaussian Processes, coined the Gaussian Process Latent Variable Model (GPLVM). The GPLVM allows for non-linear relationships between the observed and latent data space by replacing the similarity matrix in PPCA with a kernel acquired from the GP framework. In the articles, Lawrence describes the theoretical basis for the model, as well as its close connections to kernel PCA and multidimensional scaling (MDS). The GPLVM model is then applied to multiple large high-dimensional datasets and compared to other similar models.

The results show that the GPLVM model is superior in accuracy but suffers from cost due to its $\mathcal{O}(N^3)$ complexity. This prones the author to develop a sparse GPLVM model alternative. This algorithm proves to be more practical, but shows a degradation in performance on the selected datasets.

In this project we implement the (full) GPLVM algorithm using conjugate descent and apply it to the same Oilflow and USPS Digit data as used in [1], as well as to three additional high-dimensional datasets (see appendix for details about each dataset). We compare the GPLVM model to PCA, Kernel PCA, MDS and t-SNE (T-distributed Stochastic Neighbor Embedding).

In §2 we mention a couple of dimensionality reduction techniques related to GPLVM, as well as some current state-of-the-art DR techniques. §3 serves as a brief theoretical overview of GPLVM while §4 describes the implemented methods. In §5 we present the results and analyze them directly before giving a more detailed discussion in §6. Finally, §7 includes some suggestions for expansions of the project.
A list of used datasets, accompanied with source links and short descriptions can be found in the Appendix A, while Appendix B contains additional plots which might prove interesting to the reader.

2 Related Work

Among DR techniques, GPLVM stands out as one that manages to propagate a probabilistic distribution through a non-linear mapping. In that way it tries to bridge the gap between linear and non-linear DR techniques, taking the best of both sides. In the framework of GPLVMs, principal component analysis (PCA) is presented as a specific version of a GPLVM with a linear kernel [2].

In the 2004 paper [1] Lawrence compares GPLVM to stochastic neighborhood embedding (SNE) by noting that (1) could be transformed into a form of the Kullback-Leibler divergence, which is a measure of discrepancy between two probability distributions that SNE is trying to minimize [4]. This gave rise to T-distributed SNE (t-SNE), a non-linear dimensionality reduction technique which focuses on preserving local neighborhood information. Its key features are cluster preservation and cluster separation, which means that it puts an emphasis on keeping the local neighborhoods it thinks are in the same natural group tighter together when doing DR and also keeping distinct clusters away from one another in the embedding, something that will be clear in our results involving t-SNE. Both of these features were proved mathematically in [7]. Considered a state-of-the-art dimensionality reduction method, t-SNE is often also compared to uniform manifold approximation and projection (UMAP)[8]. UMAP makes assumptions on the high-dimensional data that it is (i) uniformly distributed on Riemannian manifold; (ii) the Riemannian metric is locally constant (or can be approximated as such); (iii) the manifold is locally connected. "From these assumptions it is possible to model the manifold with a fuzzy topological structure. The embedding is found by searching for a low dimensional projection of the data that has the closest possible equivalent fuzzy topological structure."[10]. We did not use UMAP in this work, however, because it is not as theoretically related to GPLVM as other techniques.

3 Theory

Given a set of centred D-dimensional data $\boldsymbol{Y} = [\boldsymbol{y}_1, ..., \boldsymbol{y}_N]^T$, we wish to project this data onto a subspace, which allows us to visualize data and facilitates separation of clusters within the data. This can be done through a mapping allowing us to express each $\boldsymbol{y}_n$ through a non-linear transformation f of its associated latent variable $\boldsymbol{x}_n$ by

$$\boldsymbol{x}_n = f(\boldsymbol{y}_n) + \beta^{-1}\boldsymbol{I}$$

Here f has an imposed Gaussian Process prior with mean $\boldsymbol{0}$ and a Radial Basis Function (RBF) kernel as covariance function, i.e. $f \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K})$, where the matrix $\boldsymbol{K}$ has entries $k_{n,m} = \alpha \exp\left(-\frac{\gamma}{2}(\boldsymbol{x}_n - \boldsymbol{x}_m)^T(\boldsymbol{x}_n - \boldsymbol{x}_m)\right) + \delta_{nm}\beta^{-1}$. Here δ_{nm} is the Kroenecker delta and β is the precision parameter of the noise term. This is the key step that allows us to use the non-linearity of Gaussian processes to create mappings from a latent space, $\boldsymbol{X} \subset \mathbb{R}^q$, to an observed data-space, $\boldsymbol{Y} \subset \mathbb{R}^d$. This is achieved by replacing the inner product kernel in the Probabilistic PCA model with a covariance function that allows for non-linear functions (in our case RBF). It

can be shown that marginalizing over the parameters $\boldsymbol{f}$ or the latent variable $\boldsymbol{X}$ are equivalent [2].

We are thus looking to maximize the log likelihood of

$$\begin{aligned} p(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{\theta}) &= \int p(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{f}, \boldsymbol{\theta}) p(\boldsymbol{f}) d\boldsymbol{f} \\ &= \frac{1}{(2\pi)^{\frac{DN}{2}} |\boldsymbol{K}|^{\frac{D}{2}}} \exp\left(-\frac{1}{2} \mathrm{tr}(\boldsymbol{K}^{-1} \boldsymbol{Y} \boldsymbol{Y}^T)\right), \end{aligned}$$

Here $\boldsymbol{\theta} = \{\alpha, \beta, \gamma\}$, with α and γ being the parameters of the RBF kernel and β the precision parameter of the random noise added to the observations. The corresponding log-likelihood then becomes

$$\mathcal{L} = -\frac{DN}{2} \ln 2\pi - \frac{D}{2} \ln |\boldsymbol{K}| - \frac{1}{2} \mathrm{tr}(\boldsymbol{K}^{-1} \boldsymbol{Y} \boldsymbol{Y}^T). \tag{1}$$

Because of using a non-linear kernel as covariance function, there is no tractable closed form for the optimal value of the parameters, $\{\boldsymbol{\theta}, \boldsymbol{X}\}$. We thus resolve to the gradient descent method to optimize $\mathcal{L}$ w.r.t. the parameters.

By differentiating $\mathcal{L}$ with respect to $\boldsymbol{K}$ we obtain the gradients necessary to apply gradient descent to optimize the expression.

The gradients can be found by combining

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{K}} = \boldsymbol{K}^{-1} \boldsymbol{Y} \boldsymbol{Y}^T \boldsymbol{K}^{-1} - \boldsymbol{D} \boldsymbol{K}^{-1} \tag{2}$$

with $\frac{\partial \boldsymbol{K}}{\partial \boldsymbol{x}_{n,j}}$ via the chain rule. This is done through the use of the function grad from the `autograd` library.

4 Method and implementations

Our implementation of the GPLVM algorithm is similar to that of [2] with the difference that ours uses gradient descent instead of scaled conjugate gradient (SCG) as an optimization method. This was due to time constraints and the main focus of this project not being on optimization methods. The usage of gradient descent will affect the execution time due to doing a line-search, which SCG avoids. Another consequence is that we now face the issue of determining the learning rates. In [1] and [2], these are determined automatically by the scaling of the steps in the SCG algorithm. Learning rates are usually found by empirical studies or sieving, but because of complexity we were not able to optimize the learning rates for each data set. We chose to use values used in an implementation of GPLVM on the Oilflow dataset found at [9]: 10^{-6} for $\boldsymbol{X}$ and α, 10^{-15} for β and 10^{-7} for γ. These learning rates were used for all data sets and may have affected performance and convergence rate since they are not optimized for these datasets but only for Oilflow.

In order to find good candidates for initializing the algorithm on the various datasets, we implemented and applied sieving and ran the algorithm for $T = 10$ iterations over a small (due to complexity) $5 \times 5 \times 5$ grid of values for α, β and γ, and chose the values which gave the highest log-likelihood values. These values were then chosen as initializations for the algorithms.

The latent variable $\boldsymbol{X}$ was initialized by standard PCA. Each dataset was then run for $T = 1000$ iterations which produced estimates of the optimal parameters $\boldsymbol{X}$, α, β and γ.

To objectively evaluate the quality of the visualisations, we classified each data point according to the class of its nearest neighbour in the two-dimensional latent space supplied by each method. For some algorithms parameter selection was required. The parameters were randomly selected, while some were determined by sieving. For HAR and USPS dataset, the GPLVM was run on a third of the complete dataset due to very high computational time.

The implementation is presented in the psuedo-code scheme `Algorithm` 1.

5 Results and analysis

Table 1 shows the accuracy score i.e mean accuracy for a range of datasets by different algorithms (data was split to a $\frac{2}{3}$ for training and $\frac{1}{3}$ for testing).

As seen in Table 1, the full GPLVM model outperforms PCA, kernel PCA and MDS on Oilflow and Vowels data, while showing similar results on the HAR and Wine datasets. Performance on USPS Digits dataset is

Algorithm 1 Implementation of GPLVM model with Gradient Descent

Require: *lowerDim*, *maxIter*, $\boldsymbol{Y}, \alpha_0, \beta_0, \gamma_0$ and $\boldsymbol{\delta}$ (learning rates)

function GPLVMFIT
  Initialize $\boldsymbol{X}_0$ ▷ Using {`PCA` or `ISOMAP`} with argument *lowerDim*, according to discussion
  Let $\boldsymbol{\theta}_0 = [\boldsymbol{X}_0, \alpha_0, \beta_0, \gamma_0]$ ▷ With $\theta_{i,t} = \boldsymbol{\theta}_t[i]$
  for $t \leftarrow 1$ to maxIter **do**
    for $i \leftarrow 0$ to size($\boldsymbol{\theta}_0$) **do**
      $\theta_{i,t} = \theta_{i,t-1} + \delta_i \cdot \text{grad}_i(\mathcal{L}(\boldsymbol{Y} \mid \boldsymbol{\theta}_{t-1}))$ ▷ $\text{grad}_i(\mathcal{L}(\boldsymbol{Y} \mid \boldsymbol{\theta}_{t-1})) = \frac{\partial}{\partial \theta_i}\mathcal{L}(\boldsymbol{Y} \mid \theta_{i,t-1})$
    end for
  end for
  return $\boldsymbol{\theta}_{\text{maxIter}}$
end function

marginally better than previously mentioned models, but is considerably outperformed by t-SNE, which scores an accuracy score of 96% or higher on all data sets except the Wine dataset. Our implementations of PCA, Kernel PCA and (metric) MDS show similar accuracy results on the Oilflow data set as in [2], however the article's results were only based on a subset of 100 data points.

	PCA	**GPLVM**	**Kernel PCA**	**MDS**	**t-SNE**
Oilflow	0.82	0.96	0.80	0.80	0.98
Vowels	0.58	0.92	0.57	0.68	0.96
HAR (UCI)	0.67	0.62*	0.66	0.64	0.96
Wine (UCI)	0.71	0.72	0.72	0.71	0.71
USPS Digits	0.48	0.57*	0.48	0.49	0.96

Table 1: Score (in fractions) of different methods when using the latent space for nearest neighbour classification in the latent space. * tested on $\frac{1}{3}$ of the training dataset.

The RBF-based GPLVM's latent space in Figure 1 shows results which are clearly superior (in terms of separation between the different flow phases and accuracy) to those achieved by the linear PCA model on the Oilflow dataset.

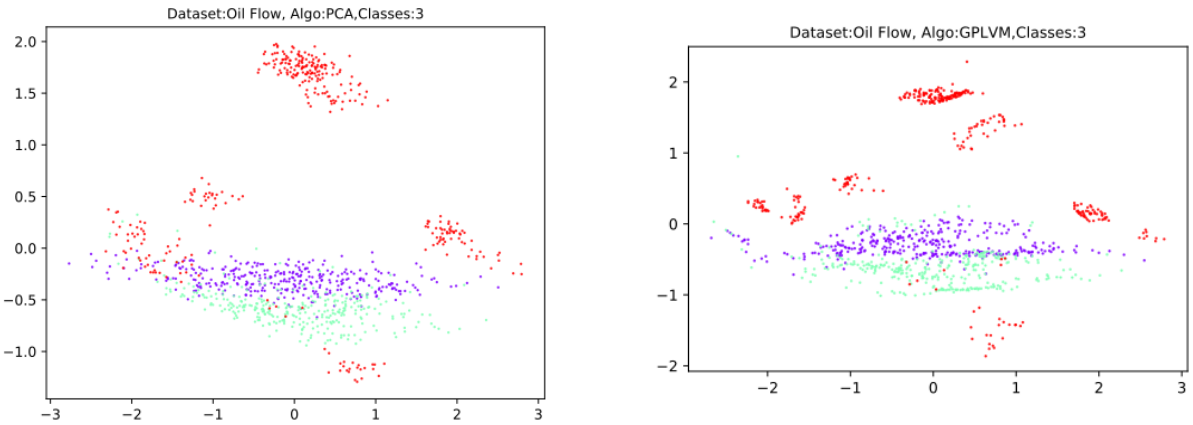

Figure 1: Visualisation of the Oilflow data with (a) PCA (a linear GPLVM) and (b) A GPLVM which uses an RBF kernel.

For the Oilflow dataset we achieve similar results as in [2] in terms of error percentage. The 96% accuracy is comparable to 99.9% achieved by Lawrence. The resulting manifold has a slightly different visual appearance compared to his, which we attribute to differing parameter initialization.

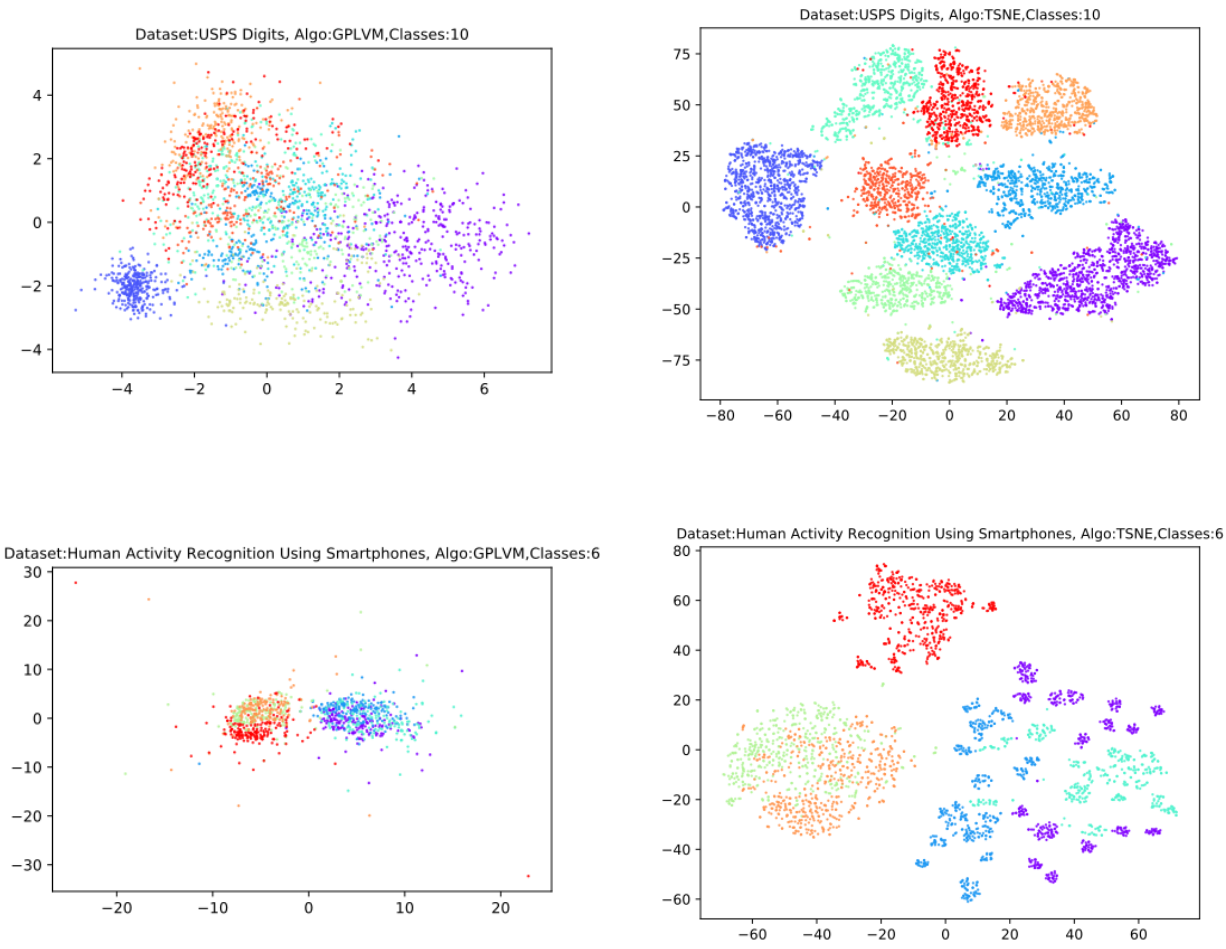

Figure 2: Visualisation of GPLVM(left), t-SNE(right) on higher dimension datasets (a) USPS Digits (top) (b) Human Activity Recognition Using Smartphones (bottom).

Contrary to the results on the Oilflow data, the results on the dataset of handwritten digits (approx. 2500 data points) show a significant decrease in performance compared to the results presented in [2]. With the full GPLVM model, we achieve an accuracy of 57%, while Lawrence achieves a 93.1% accuracy using the Sparse GPLVM model - a model they conclude shows a degradation in performance compared to the full model. This might be due to poor initialization of $\boldsymbol{X}$ or the other parameters. Lawrence argues that sometimes PCA offers a poor initialization of $\boldsymbol{X}$, causing it to get stuck in a local minimum. Indeed, in our implementation PCA only scores an accuracy of 57% as compared to Lawrence's 74%. Adhering the authors suggestion we therefore applied ISOMAP to provide an initialization of $\boldsymbol{X}$, but with a degredation in performance as a result. This suggests some other issue with either initialization or implementation as a whole, but consistent results of our implementation on other datasets suggest the former. In order to validate Lawrence's claim on ISOMAP, we applied GPLVM initialized with ISOMAP on the HAR dataset. We observed a 62% accuracy with PCA initialization and 71.4% with ISOMAP initialization, providing some support for the claim.

As mentioned in §2, t-SNE has been mathematically proven to give good cluster separation, ensuring that the preserved clusters from the high-dimensional space do not overlap in the low-dimensional space. This can be noticed in Figure 2 as well as in Table 1.

6 Discussion

6.1 Model Sparsification

An analysis of the complexity of the algorithm shows that each gradient step requires an inverse of the kernel matrix, an $O(N^3)$ operation that renders the algorithm impractical for large datasets. A practical approach, as suggested by the author, is to use model sparsification. By exploiting a sparse approximation to the full Gaus-

sian process, it is usually possible to reduce the computational complexity to $O(k^2N)$,where k is the number of points retained in the sparse representation. The author uses informative vector machine (IVM) to select an active set and an inactive set according to the reduction of the posterior's process entropy. This, however, resulted in degradation of performance because the resulting posterior distribution over mappings depended only on the active set which was subject to a change in each iteration of the optimization process. This re-selection resulted in fluctuations in the objective function. It was therefore difficult to determine when convergence had occurred.

In his subsequent paper [11], Lawrence focused on approximating the covariance function with a low rank approximation. Quinonero Candela and Rasmussen[12] also provided a consistent terminology for these approximations known as *QR Terminology*. These approximations included augmenting the function values at the training points, $f \subset \mathbb{R}^{N\times d}$, and the function values at the test points, $f^* \subset \mathbb{R}^{\infty\times d}$, by an additional set of variables, $U \subset \mathbb{R}^{k\times d}$, known as 'inducing variables' which are selected by the user. Deterministic Training Conditional (DTC) is one such approximation in QR Terminology which replaces the true training conditional with a deterministic approximation. Snelson and Ghahramani [13] also suggested the use of the Fully Independent Training Conditional (FITC) approximation and optimization of the inducing inputs jointly with the parameters of the covariance function. By jointly optimizing over $\boldsymbol{X}$, $\boldsymbol{X}_u$ and $\boldsymbol{\theta}$, convergence can be monitored in a straightforward manner and the issue faced by IVM are avoided (reselection of active set from $\boldsymbol{X}$).
We used the python package GPy to compare the results of GPLVM and Sparse GPLVM on the Oilflow dataset (see Table 2). We used 30 inducing points and performed 1000 iterations of each algorithm. The sparse algorithm took 89% less time to complete than the full one. The KNN score for standard GPLVM was similar to our implementation (see Table 1), the score for Sparse GPLVM was relatively lower as expected due to functional approximations.

	GPLVM	**Sparse GPLVM**	**Bayesian GPLVM**
Running Time (s)	314.70	32.60	119.33
KNN Score	0.98	0.94	0.97

Table 2: Comparison of GPLVM, Sparse GPLVM and Bayesian GPLVM on OilFlow dataset from GPy package.

6.2 Variational Bayesian GPLVM

The Variational Bayesian GPLVM approach presented in [14] is an extension of standard GPLVM that is robust to overfitting and its ability to automatically select the dimensionality of the nonlinear latent space. Moreover, it is generic and flexible to be extended for other problems such as Gaussian process regression with missing data and semi-supervised Gaussian Processes.
The Variational Bayesian approach is intractable because of the non linearity present in Latent space inside the co-variance matrix K_{NN},hence Lawrence introduced auxiliary variables m which are $m \ll N$. This makes the lower bound tractable and also reduce the computation cost of it to $O(Nm^2)$ in comparison to $O(N^3)$ of standard GPLVM that enables to work on very high dimensional data. The improvement in running time can be inferred from Table 2: the running time is almost half of the standard GPLVM with an equivalent K-NN score. The selection of dimensions is achieved by adding a different length scale per input dimension (weights) in the RBF covariance function. The Automatic Relevance Determination (ARD) procedure makes the weight w_k of the dimension to almost 0 which cannot be achieved with the standard approach. The improvement can be seen in Figure 3. The inter cluster distance is much larger with denser clusters, and grey scale representing the precision is much higher in 3b in comparison to 3a.

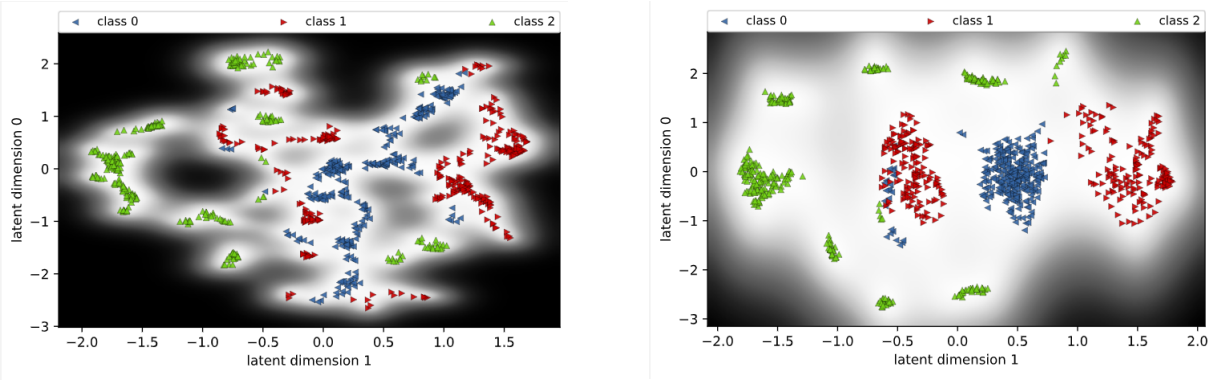

Figure 3: Plot (a) shows the visualization found by standard sparse GPLVM. (b) shows the visualization found by Bayesian GPLVM, on the Oilflow dataset using GPy package.

7 Conclusion and future work

To conclude, we have managed to replicate the results from [1] and [2] and compared these to some other DR techniques. We have found that GPLVM indeed outperforms PCA, Kernel PCA and MDS on most datasets, but is dwarfed by the results of the state-of-the-art method t-SNE, both in accuracy and speed. We also find that the sparse GPLVM model is a viable alternative when complexity is an issue, but with a degradation in performance as a result. Our exploration on Variational (Bayesian) GPLVM suggest that this might be a better alternative for sparsification, as the running time is half that of regular GPLVM, and its accuracy close to that of the full model.

7.1 Alternative error measures

In this project we have relied on k-NN with $k = 1$ as our quality measure when assessing the different embedding techniques. While 1-NN is a very simple measure to implement and understand, it is also quite crude in the sense that one can easily conceive of situations in which it would give us non-informative and sometimes even misleading results (think of a situation where low-dimensional data points come more or less in pairs with points from their original cluster, surrounded with pairs of points from different clusters). A possible addition to our comparison would be to use more quality measures than just 1-NN. A pair of quality measures that were used in [6] are *trustworthiness* and *continuity*. We give the formal definitions below.

Definition 1 *Analogous to [5], we define here the* trustworthiness *of an embedding* $\boldsymbol{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_N}\} \subset \mathbb{R}^q$ *of a high-dimensional dataset* $\boldsymbol{Y} = \{\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_N}\} \subset \mathbb{R}^d$ *with* k *degrees of freedom as*

$$T(k) = 1 - \frac{2}{Nk(2N - 3k - 1)} \sum_{i=1}^{N} \sum_{j \in U_i^{(k)}} (\max(0, r(i,j) - k)),$$

where $r(i,j)$ *represents the rank*[1] *of the low-dimensional datapoint* $\boldsymbol{x_j}$ *according to the pairwise distances between the low-dimensional datapoints. The set* $U_i^{(k)}$ *contains the points that are among the* k *nearest neighbors to datapoint with index* i *(denoted* $\boldsymbol{y_i}$ *when in* $\mathbb{R}^d$ *and* $\boldsymbol{x_i}$ *when in* $\mathbb{R}^q$*) in the low-dimensional space but not in the high-dimensional space.*

Definition 2 *Analogous to [5], we define the* continuity *of an embedding* $\boldsymbol{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_N}\} \subset \mathbb{R}^q$ *of a high-dimensional dataset* $\boldsymbol{Y} = \{\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_N}\} \subset \mathbb{R}^d$ *with* k *degrees of freedom as*

$$C(k) = 1 - \frac{2}{Nk(2N - 3k - 1)} \sum_{i=1}^{N} \sum_{j \in V_i^{(k)}} (\max(0, \hat{r}(i,j) - k)),$$

where $\hat{r}(i,j)$ *represents the rank of the high-dimensional datapoint* $\boldsymbol{y_j}$ *according to the pairwise distances between the high-dimensional datapoints. The set* $V_i^{(k)}$ *contains the points that are among the* k *nearest neighbors to datapoint with index* i *in the high-dimensional space but not in the low-dimensional space.*

These two quality measures were used by van der Maaten, Postma, and van den Herik in their GPLVM when comparing a large number of DR techniques [5]. To summarize them, we can say that $T(k)$ is penalized if distant points become neighbors while $C(k)$ is penalized if neighboring points become distant.

[1] By rank we mean the rank based on closeness to the point $\boldsymbol{x_i}$; if $\boldsymbol{x_j}$ is the nearest neighbor to $\boldsymbol{x_i}$, then $r(i,j) = 1$. Similarly, if $\boldsymbol{x_i}$ is the fifth nearest neighbor to $\boldsymbol{x_i}$, then $r(i,j) = 5$.

References

[1] Lawrence, N.D., 2004. Gaussian process latent variable models for visualisation of high dimensional data. In Advances in neural information processing systems (pp. 329-336).

[2] Lawrence, N., 2005. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. Journal of machine learning research, 6(Nov), pp.1783-1816.

[3] Mardia, K.V., 1978. Some properties of clasical multi-dimesional scaling. Communications in Statistics-Theory and Methods, 7(13), pp.1233-1241.

[4] Hinton, G.E. and Roweis, S.T., 2003. Stochastic neighbor embedding. In Advances in neural information processing systems (pp. 857-864).

[5] Van Der Maaten, L., Postma, E. and Van den Herik, J., 2009. Dimensionality reduction: a comparative. J Mach Learn Res, 10(66-71), p.13.

[6] Maaten, L.V.D. and Hinton, G., 2008. Visualizing data using t-SNE. Journal of machine learning research, 9(Nov), pp.2579-2605.

[7] Linderman, G.C. and Steinerberger, S., 2019. Clustering with t-SNE, provably. SIAM Journal on Mathematics of Data Science, 1(2), pp.313-332.

[8] McInnes, L., Healy, J. and Melville, J., 2018. Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426.

[9] `https://github.com/sbrml/gaussian_processes/blob/master/code/gplvm.ipynb` (Accessed Jan 13th 2020)

[10] `https://umap-learn.readthedocs.io/en/latest/` (Accessed Jan 13th 2020)

[11] Lawrence, N.D.. (2007). Learning for Larger Datasets with the Gaussian Process Latent Variable Model. Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, in PMLR 2:243-250

[12] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. 2005. A Unifying View of Sparse Approximate Gaussian Process Regression. J. Mach. Learn. Res. 6 (December 2005), 1939,1959

[13] Edward Snelson and Zoubin Ghahramani. 2005. Sparse Gaussian processes using pseudo-inputs. In Proceedings of the 18th International Conference on Neural Information Processing Systems (NIPS'05). MIT Press, Cambridge, MA, USA, 1257,1264.

[14] Damianou, Andreas & Titsias, Michalis & Lawrence, Neil. (2014). Variational Inference for Uncertainty on the Inputs of Gaussian Process Models.

A Datasets

Phase Oil Data

This is synthetic data modelling non-intrusive measurements on a pipe-line transporting a mixture of oil, water and gas. The data has 1000 measurements each with 12 dimensions and three class labels.
https://inverseprobability.com/3PhaseData.html .

Handwritten Digits USPS dataset

The dataset has 7291 train and 2007 test images. The images are 16*16 grayscale pixels so 256 dimensions.
https://www.kaggle.com/bistaumanga/usps-dataset/metadata

Wine Data Set

These data (178) are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.
https://archive.ics.uci.edu/ml/datasets/wine

Human Activity Recognition Using Smartphones Data Set

Human Activity Recognition database built from the recordings of 30 subjects performing activities of daily living (ADL) while carrying a waist-mounted smartphone with embedded inertial sensors.There are 561 attributes, 10299 samples.)
https://scikit-learn.org/0.19/datasets/olivetti_faces.html

The Vowel Data Set

In the original vowel data set, the 11 different words, with each one representing a vowel sound, correspond to 11 different clusters. There are 528 training observations. There are 10 dimensions.
https://rdrr.io/cran/clues/man/Vowel.html

B Additional Plots