Gaussian Process Latent Variable Models for Dimensionality Reduction

Josef Haddad, David Norrman,
Carl Piehl, Svante Sörberg

January 2020

Abstract

We have implemented a Gaussian Process Latent Variable Model (GP-LVM), as described by Lawrence in [4] and used it for dimensionality reduction on a standard dataset. We use three different methods (IVM, k-means and random sampling) for sparsification to speed up the training of the GP-LVM. To test these we used the same data set used in [4] for simpler comparison. The results showed that there was not much difference of GP-LVM in comparison to PCA and the addition of sparsification worsened the dimensionality reduction.

1 Introduction

High-dimensional data is commonplace in many machine learning and data science applications. As dimensionality increases, so typically the time required to train models and the amount of data needed, due to the curse of dimensionality [1, 5]. It is often the case that high-dimensional data can be projected onto a lower-dimensional space that captures the most important aspects of the data [5]. This can be interpreted as a task of recovering latent, low-dimensional, variables in $\mathbb{R}^q$ from observed, high-dimensional, data in $\mathbb{R}^D$ $(D > q)$. A standard technique is Principal Component Analysis (PCA) [5]. An interpretation of the classic PCA is that each latent point $\mathbf{x}_i$ generates its corresponding observed data point $\mathbf{y}_i$ according to $p(\mathbf{y}_i \mid \mathbf{x}_i) = \mathcal{N}\left(\mathbf{y}_i \mid \mathbf{W}\mathbf{x}_i + \boldsymbol{\mu}, \sigma^2\mathbf{I}\right)$ as $\sigma^2 \to 0$.[1] If σ^2 is allowed to be any positive constant, the method is instead called Probabilistic Principal Component Analysis (PPCA) [5, 7]. In PPCA, one optimizes the parameters. One might instead consider optimizing the latent variables themselves. The gradient of the likelihood w.r.t. the latent variables is

$$\frac{\partial L}{\partial \mathbf{X}} = \mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^T\mathbf{K}^{-1}\mathbf{X} - D\mathbf{K}^{-1}\mathbf{X}$$

where $\mathbf{K} = \mathbf{X}\mathbf{X}^T + \sigma^2\mathbf{I}$ [4]. We will explore an extension of this model - the Gaussian Process Latent Variable Model, or GP-LVM, and see how different methods for sparsification can improve its time complexity.

2 GP-LVM

The GP-LVM can be considered an extension of PPCA where one optimizes the latent variables, but where the matrix $\mathbf{K}$, which from now on will be called the **kernel matrix** is not required to be linear. The optimization problem is still tractable. The gradient of the likelihood w.r.t. the latent points can be derived by taking the gradient w.r.t. the kernel matrix and combining it with the gradient of the kernel matrix w.r.t. the latent points through the matrix chain rule [6, eq. 137]:

$$\frac{\partial L}{\partial X_{ij}} = Tr\left[\left(\frac{\partial L}{\partial \mathbf{K}}\right)^T \frac{\partial \mathbf{K}}{X_{ij}}\right]$$

By [4, eq. 10], the gradient of the likelihood w.r.t. $\mathbf{K}$ is:

$$\frac{\partial L}{\partial \mathbf{K}} = \mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^T\mathbf{K}^{-1} - D\mathbf{K}^{-1}$$

As long as $\frac{\partial \mathbf{K}}{X_{ij}}$ is computable, the latent points can be optimized to maximize the likelihood. Additionally, the parameters of the kernel function can also be optimized jointly with the latent points. As Lawrence, we used the method of Scaled Conjugate Gradients (SCG) to optimize latent points and kernel parameters. SCG is implemented in the Python library *Paramz*.
In the original paper, Lawrence used an RBF kernel with four parameters:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \theta_{rbf} exp\left[-\frac{\gamma}{2}(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)\right] + \theta_{bias} + \theta_{white}\delta_{ij}$$

Since our primary purpose is to investigate data sparsification methods, We have employed a simpler RBF kernel with only two parameters:

$$k(\mathbf{x}_i, \mathbf{x}_j) = exp\left[-\frac{\gamma}{2}(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)\right] + \theta_{bias}$$

Because the likelihood is non-linear w.r.t. the latent points and kernel parameters, there may be many local optima. For this reason, it is worth considering good initialization methods. As Lawrence, our initial guess for the latent points are obtained from regular PCA. For the kernel parameters, we provide the starting guesses $\gamma = 1, \theta_{bias} = 0$

[1] The latent representation can be obtained by maximizing the likelihood w.r.t. the parameters.

3 Sparsification

A drawback of the GP-LVM model is that for N data points, the gradient evaluations are $O(N^3)$ because they involve inversion of the $N \times N$ kernel matrix $\mathbf{K}$. To mitigate this problem, it is useful to employ some method of sparsification: using fewer data points. In a sense, this is similar to dimensionality reduction techniques. Instead of finding a lower-dimensional subspace that accounts for as much information as possible, one finds a subset of data points that accounts for as much information as possible. This subset is called the *active set* – the remaining data points belong to the *inactive set*.

One can maximize the likelihood of the active set w.r.t. the kernel parameters and latent points, which is $O(d^3)$ for an active set of size d. By [4, eq. 12], the likelihood for a point $\mathbf{y}_j$ in the inactive set is then given by a multivariate normal distribution with mean and covariance that are functions of the kernel matrix of the (latent) active set, and the kernel function applied to the latent points in the active set and the latent inactive point $\mathbf{x}_j$. Because optimization of $\mathbf{x}_j$ in the inactive set only depends on the active set and $\mathbf{y}_j$ (and no other points in the inactive set), optimization of the N latent points can be achieved quicker than $O(N^3)$. We used the Nelder Mead method for optimizing all $\mathbf{x}_j$.

3.1 Informative Vector Machine

The Informative Vector Machine (IVM), which is the model used by Lawrence in his original implementation, performs a greedy sequential selection of points according to their reduction of the posterior entropy [4]. In each iteration, the algorithm calculates the differential entropy score for all of the points in the inactive set. The point which maximizes this score is selected and moved to the active set. This is repeated until the active set is of desired size. The idea is that the entire data set can be represented by the points which have the largest impact on posterior entropy. The active points fill a purpose similar to that of the support vectors in an SVM. An SVM represents its decision boundary by only using a fraction of the training data by picking the points which have the largest impact on the boundary. In the context of data sparsification, this means picking the data points which are most representative of the entire data set. The model has a time complexity of $O(nd^2)$, this makes it quite efficient when $n >> d$.

3.2 K-means

The idea behind using k-means for data sparsification is that it should be able to capture a set of points that can represent the whole data set [3]. The selected subset can then be used as candidate for the active set and since these data points don't necessarily exist in the original data set, all the true data points are put into the inactive data set. A reason why we choose k-means instead of only randomly selecting a subset to use as the active data set is that k-means aims at minimizing the sum of squared distances for the data points to the closest cluster centroid [3]. This means that with sufficient number of cluster centroids, even small clusters consisting of very few data points in relation to the larger clusters in the data should be represented by cluster centroids, while this probably will not be the case when we randomly select data points to be in the active data set. We used Lloyd's algorithm [8] and the euclidian distance between data points and cluster centrids to implement k-means. The time complexity of Lloyd's algorithm is $O(ndK)$ where n is the number of data poins, d is the dimension of the data points, and K is the number of clusters. This complexity is very similar to that of the IVM. The max number of iterations in Lloyd's algorithm was set to a constant.

4 Application on datasets

We made our own implementations of the k-means[3] and IVM[2] algorithms. We tested them on a 2-dimensional toy data set to see if they could be good candidates for sparsification, the results can be seen in figure 1.

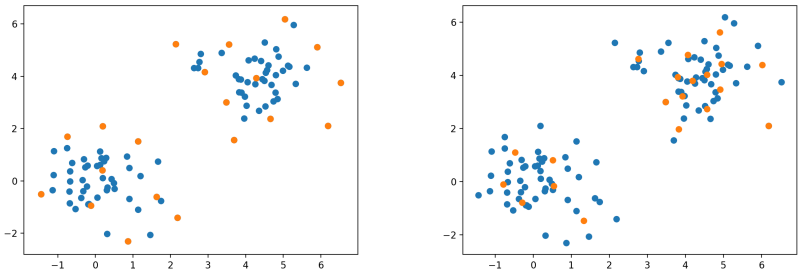

Figure 1: Here we have run IVM and k-means on a two-dimensional data set consisting of two clusters, to see if their sparse approximation captures the data well. The left plot shows the results for IVM, and the right plot shows the results for k-means. The biggest difference seems to be that, while IVM spreads its points evenly, k-means has more points from one of the clusters.

We used the oil flow data set used in the original GP-LVM paper [4]. To compare the sparsification models we first ran regular GP-LVM on the data and visualized the results. Then we used sparse GP-LVM, with IVM, k-means and random selection to see if they gave similar visualizations. We did this for 100 and 500 data points, using half of the data as active each time. For k-means we modified the original algorithm in[4]. Since the data points returned by the sparsification algorithm are not actual points in the real data set we only use them to fit the parameters, then the entire data set is regarded as inactive and optimized with respect to the latent points. The results are visualized in figure 2 and figure 3 below.

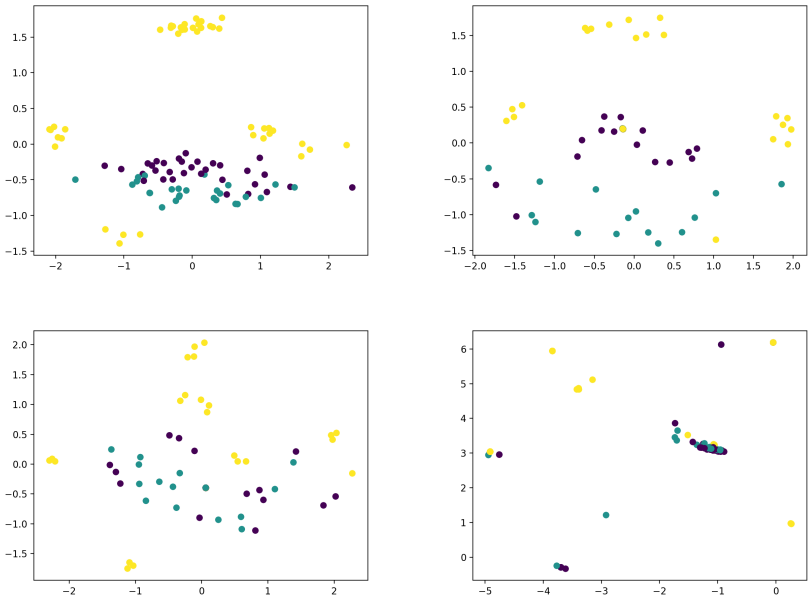

Figure 2: These are the results for the oil flow data with 100 data points. Top left: using full GP-LVM on the entire data set. Top right: sparse GP-LVM with 50 randomly selected data points in each iteration. Bottom left: sparse GP-LVM with IVM, using 50 active points. Bottom right: sparse GP-LVM with k-means, using $K = 50$.

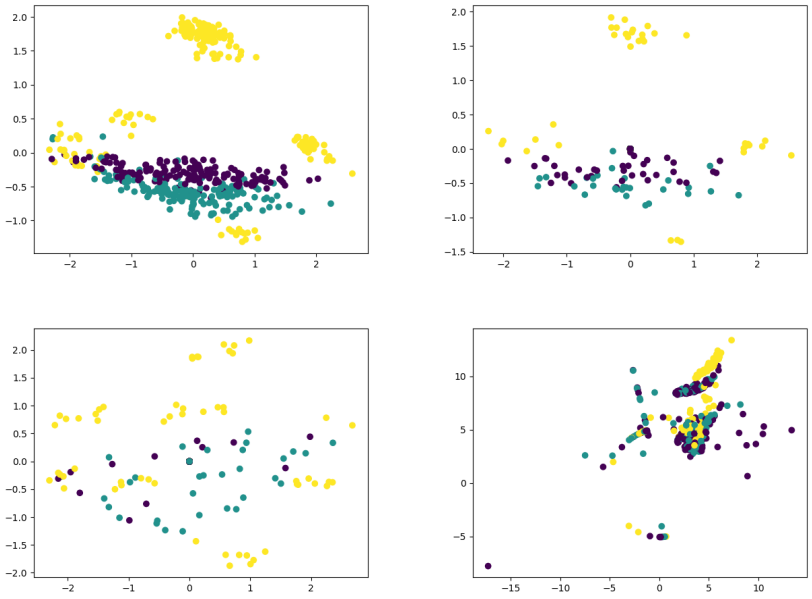

Figure 3: These are the results for the oil flow data with 500 data points. Top left: using full GP-LVM on the entire data set. Top right: sparse GP-LVM with 100 randomly selected data points in each iteration. Bottom left: sparse GP-LVM with IVM, using 100 active points. Bottom right: sparse GP-LVM with k-means, using $K = 50$.

As seen in the top left plots, the results from running full GP-LVM on the oil flow data did not yield the same results as those by Lawrence in [4]. The results are more similar to those of regular PCA, as seen in Figure 1(a) in [4]. In the bottom left images we see the results with sparse GP-LVM, using IVM to select the active set. We can see that the plot looks much more sparse than for the full model, this is due to some points being projected on the same spot, there are also several points for which the optimization function failed. In the top right image the random selection shows results similar to those of the IVM version, again several points are missing due to being projected on top of each other or the optimization function failing. Finally, in the bottom right image we have the k-means version, which seem very different from the full GP-LVM model. It should also be noted that k-means showed very unreliable results in general.

5 Discussion

From the plots it seems like the dimensionality reduction is better when using IVM for creating an active data set in comparison to random selection and k-means. This could indicate that IVM is better for representing the whole data set for optimizing the kernel-hyperparameters in comparison to the other two. It is unclear to us why k-means is as unreliable as it is, but one theory we have is that outliers get too much influence due to the high number of clusters used, which means that they get one cluster to their own, while for example 100 other similar data points are only represented by one cluster centroid as well. However, still could also very well be caused by the fact that, for k-means, when we optimize the inactive points, we use the entire data set. When using IVM we optimize the inactive points and the active points separately, and the optimization of the active points is initialized with PCA. Thus a reason why IVM performs better could be that more of its data is initialized with PCA.

We did expect the clusters of the representations in latent space to be worse when dividing the data into an active and inactive set rather than optimizing all parameters using all the data available since we have more information with more data. However, we did not expect it to get worse to the extent seen in our results. Again, this is probably caused by our optimization of the inactive points not working correctly.

Another thing to bear in mind is that the data is very high-dimensional and the sample size quite small.

This makes it much harder to make sparse representations of the data. Using a lower dimensional data set could probably have been a good idea to see if it would give different results.

Our implementations of IVM and k-means seem to work quite well independently, as seen in figure 1. The poor results are, therefore, most likely caused by our GP-LVM implementation, and since the optimization of the inactive set gives especially poor performance, we believe that the main issue is the optimization of latent variables. Thus, the problem is either with our calculation of the gradient of the likelihood, w.r.t. the data points, or with the optimization function we used.

5.1 Future work

There are multiple areas in which this report can be extended, but mainly there are two parts we would have focused on. Firstly the way we optimized the inactive points taken through the sparsification methods can be looked into. Secondly, you can test more sparsification methods. Regarding the optimization of the inactive points, you can test more methods by changing the function being optimized, and by changing the optimization method used.

Extending the report further would be testing it on more data sets and expanding away from GP-LVM. It would be very interesting seeing how different methods of sparsification behaved combined with different methods of dimensionality reduction. By having access to more powerful computers you can also test the models on larger data sets, where the sparsification parts might be to better use. Implementing parallelized variants would also be an interesting way of expanding this to larger data.

Since the report by Lawrence [4] there have been a lot of advancements in machine learning. Comparing these methods with more recent methods in areas such as deep learning would also be an interesting study. Measuring factors like time complexity, code complexity, results and seeing if newer methods are worth it.

References

[1] Bishop, C. M. [2006], *Pattern recognition and machine learning*, springer.

[2] Herbrich, R., Lawrence, N. D. and Seeger, M. [2003], Fast sparse gaussian process methods: The informative vector machine, *in* 'Advances in neural information processing systems', pp. 625–632.

[3] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R. and Wu, A. Y. [2002], 'An efficient k-means clustering algorithm: Analysis and implementation', *IEEE Transactions on Pattern Analysis & Machine Intelligence* (7), 881–892.

[4] Lawrence, N. [2005], 'Probabilistic non-linear principal component analysis with gaussian process latent variable models', *Journal of machine learning research* **6**(Nov), 1783–1816.

[5] Murphy, K. P. [2012], *Machine learning: a probabilistic perspective*, MIT press.

[6] Petersen, K. B. and Pedersen, M. S. [2012], 'The matrix cookbook, nov 2012', *URL http://www2. imm. dtu. dk/pubdb/p. php* **3274**.

[7] Tipping, M. E. and Bishop, C. M. [1999], 'Probabilistic principal component analysis', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **61**(3), 611–622.

[8] Vasconcelos, C. N., Sá, A., Carvalho, P. C. and Gattass, M. [2008], Lloyd's algorithm on gpu, *in* 'International Symposium on Visual Computing', Springer, pp. 953–964.