Advanced Machine Learning Algorithms GMR & GPR, a comprehensive study

Nathan Müller (271786)
School of Engineering (STI), Robotics (MT-RO)
École polytechnique fédérale de Lausanne (EPFL)
Lausanne, Switzerland
Email: nathan.muller@epfl.ch

Louis Piotet (260496)
School of Engineering (STI), Robotics (MT-RO)
École polytechnique fédérale de Lausanne (EPFL)
Lausanne, Switzerland
Email: louis.piotet@epfl.ch

***Abstract*—In this paper, we compare two renowned regression techniques commonly used in Machine Learning: Gaussian Mixture Regression (GMR) and Gaussian Process Regression (GPR).**

Our objective is to discuss the similarities and differences of these algorithms as well as expose their strengths and weaknesses using toy datasets. Finally, both regression techniques will be applied on a non-linear and high dimensional dataset with medium noise that concerns the forward kinematics of an 8 joint robotic arm [12].

We expect GPR to give a more accurate prediction than GMR at the cost of being more computationally expensive since it retains all datapoints for retrieval. GMR usually needs less parameters to estimate the regression curve since the number of dimensions is most often much smaller than the number of datapoints.

The result followed our expectations with this dataset. The coefficient of determination was of 0.91 and 0.8 for GPR and GMR respectively.

To conclude, GPR usually gives a more accurate result than GMR at the cost of occupying more memory and being more expensive computationally.

I. Introduction

There is a growing interest in machine learning to design powerful algorithms performing high dimensional nonlinear regression. These algorithms are interesting to use when the relations between the inputs and outputs of the system cannot be derived analytically because the system considered is too complex, or due to a lack of knowledge about the system. In this case, these algorithms can build a model for the observed data. Another interesting property is that many of these algorithms give a measure of the variance of the data around its trend and are hence able to give a measure of the noise. Furthermore, some of them are capable to measure the uncertainty of the model which is useful if one wants to know how much the prediction can be trusted in certain regions.

Some regression techniques are also capable to predict the trend of the data, outside of the learned range. Getting additional knowledge out of a limited set of samples may be extremely valuable. Typical prediction tasks include predicting stock exchange or a robot pose. A practical example may be that one collects some hard to obtain measures, as well as others, trouble-free that might explain it. If it is possible to fit a regressive model with the output variable being the one that is problematic, it could be possible to estimate its value on larger regions.

In this paper, we will focus on both *Gaussian Mixture Regression* (GMR) and *Gaussian Process Regression* (GPR). Studying their properties and identifying their pros and cons.

A. Gaussian Mixture Regression

GMR consists of training a Gaussian mixture model (GMM) on the dataset. GMM is a method that fits a mixture of Gaussians to multidimensional data [1]. It usually proceeds by maximizing the likelihood of the model in an iterative manner with Expectation-Maximization (E-M) steps. The model of K Gaussians is defined for some data x:

$$p(x) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(x|\mu_k, \Sigma_k) \tag{1}$$

Where the sum of mixing coefficients (α_k), also called priors, is equal to one. The priors represent the relative importance of each Gaussian component i.e how well the Gaussian explains the dataset. The hyper parameters of GMM are the number of Gaussians (K) and the type of the covariance matrix (full, diagonal or spherical). Learning the model consist of determining the mean (μ_k) and covariance matrices (Σ_k) as well as the mixing coefficient for each Gaussian.

In the case of GMR, we have a set of inputs $X = \{x^1, ..., x^M\}$ with $x^i \in \mathbb{R}^{N_x}$ and a set of outputs $Y = \{y^1, ..., y^M\}$ with $y^i \in \mathbb{R}^{N_y}$ (the output can be multidimensional) and the model is expressed as :

$$p(x, y) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(x, y|\mu_k, \Sigma_k) \tag{2}$$

with the associated means and variance matrices:

$$\mu_k = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \Sigma_k = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix}$$

Once the model is trained, GMR estimates the desired output on a query point (x^*) by using the expectation on the condi-

tional of this variable and the variance of the conditional is called predictive variance:

$$y(x^*) = \mathrm{E}[p(y|x^*)] = \sum_{k=1}^{K} \beta_k(x^*)\mu_{y|x}^k(x^*) \tag{3}$$

$$\begin{aligned}\mathrm{Var}\left(p(y|x^*)\right) = \sum_{k=1}^{K} \beta_k(x^*)((\mu_{y|x}^k(x^*))^2 + \Sigma_{y|x}^k) - \\ (\sum_{k=1}^{K}(\beta_k(x^*)\mu_{y|x}^k(x^*)))^2\end{aligned} \tag{4}$$

with:

$$\beta_k(x) = \frac{\alpha_k \mathcal{N}(x|\mu_x^k, \Sigma_x^k)}{\sum_{k=1}^{K} \alpha_k \mathcal{N}(x|\mu_x^k, \Sigma_x^k)}$$

$$\mu_{y|x}^k(x) = \mu_y^k + \Sigma_{yx}^k(\Sigma_{xx}^k)^{-1}(x - \mu_x^k)$$

$$\Sigma_{y|x}^k = \Sigma_{yy}^k - \Sigma_{yx}^k(\Sigma_{xx}^k)^{-1}\Sigma_{xy}^k$$

The expectation is thus a non-linear combination of the expectation of each local component. The variance of GMR is modulated by the variance of each component locally and hence carries across a notion of local variance. This variance represents the uncertainty of the prediction, hence the local variance of the data around the prediction. In order to capture the uncertainty of the model, one should compute the Likelihood of the model at the query point.

B. Gaussian Process Regression

Given a set of inputs $X = \{x^1, ..., x^M\}$ with $x^i \in \mathbb{R}^{N_x}$ and a set of outputs $Y = \{y^1, ..., y^M\}$ with $y^i \in \mathbb{R}$ (the output is uni-dimensional), GPR can be derived from the probabilistic linear regression equation:

$$y = w^T x + \epsilon \tag{5}$$

Where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is some zero-mean Gaussian noise with standard deviation σ. Making the hypothesis that weights are random variables having a distribution defined by $\mathcal{N}(m(x), \Sigma_w)$ and defining the model likelihood $p(y|X, \hat{w}, \sigma_\epsilon^2)$, one can take a Bayesian approach and compute the posterior:

$$p(w|X, y) = \frac{p(y|X, \hat{w}, \sigma_\epsilon^2)}{p(y|X)} \tag{6}$$

The weights are taken as the maximum a posteriori (MAP) estimate of w, the derivation is described in [2], [4] and [3]. Applying the kernel trick to perform nonlinear regression, the prediction made by the model with M training datapoints and kernel function $k(x, x^i)$ at a query point (x^*) is defined as :

$$y(x^*) = \sum_{i=1}^{M} m(x^*) + \alpha_i k(x^*, x^i) \tag{7}$$

with : $\alpha = [K(X, X) + \sigma^2 I]^{-1}(Y^T - m(X))$, where K is the gram matrix: $K_{i,j}(X, X) = k(x_i, x_j)$

The function $m(x)$ is the mean function and is often set to zero in practice. However, if one has insights on the global trend of the data, the mean function can be designed to follow this trend. This will have a great influence on the prediction made far away from the training points.

The predictive variance of GPR at a query point is defined as follows:

$$Var(y(x^*)) = k(x^*, x^*) - k(x^*, X)[K(X, X) + \sigma^2 I]^{-1}k(X, x^*) \tag{8}$$

In contrary to GMR, this variance captures both the variance of the data around the prediction (which is here assumed fixed) and the model uncertainty. A lack of points in a certain area produces a large predictive variance.

As judging visually the performance of a regression when working with a high dimensional dataset is often not possible, the marginal likelihood $p(y|X)$ can be used to measure how well the model explains the data. The marginal negative log-likelihood is defined below:

$$\begin{aligned}-log(p(Y|X)) = \\ 0.5 * (Y^T - m(X))^T A^{-1}(Y^T - m(X)) \\ + log(|A|2\pi^n))\end{aligned} \tag{9}$$

With $A = [K(X, X) + \sigma^2 I]$

The kernel parameters as well as the mean function can be set by the user or optimised by maximising the marginal likelihood. This optimization provides a trade-off between the quality of the fit given by the quadratic term and the complexity of the model, coming from the determinant of the matrix A. A grid search can also be used to find optimal hyperparameters.

As a summary, GPR places a kernel with an associated weight (α_i) to every point in the dataset. The prediction is obtained by summing the weighted contribution of all those kernels evaluated at the query point.

C. Objectives

In this paper, we will first describe the hyperparameters of both methods and an intuition about their effects will be given using a 2D toy dataset. Next, we will compare their sensitivity to gaps, outliers and sparsity of the data. Our hypothesis is that the prediction of the model in these cases will greatly depend on the hyperparameters and their effect will be illustrated. We will then assess their sensitivity to initialisation. A comparison of their computational complexity will be presented. Finally, the two algorithms will be compared on a non linear high dimensional dataset using various metrics. Our hypothesis is that GPR should perform better since it relies on every datapoints to predict the regression curve.

II. Methods

A. Hyperparameters

Gaussian Mixture Regression requires two hyperparameters, the number of Gaussian components (K) and the shape of the covariance matrix Σ (spherical, diagonal or full). For regression, full matrices are generally used as we expect the data to be correlated, i.e have a slope that is neither zero nor infinite for a regression in 2D. The number of Gaussian component corresponds to the flexibility we give to the model to fit the data. As K greatly influence the fit, one should care about under or over-fitting.

The choice of the hyperparameters has a high influence in the number of parameters computed for the model. This number follows this formula in the case of full covariance matrices and a dataset of dimension N:

$$N_{params} = K(N\frac{(N+1)}{2} + N) + K - 1 \tag{10}$$

If the number of datapoints available is limited, a model with many components may result in poor confidence for the parameters of the models. Having at least 10 datapoints per number of parameters is a least requirement for a fair model [5]. Thus, a rule of the thumb would be :

$$M \geq 10 \cdot N_{params} \tag{11}$$

Gaussian Process Regression requires three hyperparameters: the estimated standard deviation of the signal noise (σ_e) the kernel function and the mean function ($m(x)$). σ_e expresses the supposed noise in the input signal. If it is approaching zero, GPR is being told that there's high confidence that each datapoint is representative of the regressive function. The choice of the kernel function will then add in one or multiple hyperparameters to the model. Numerous kernels are available such as the Radial Basis Function (RBF), Exponential Squared Sine and others. However, a composition of various kernels is often used in practice as they tend to offer more advanced features [7]. Finally, when each dimension has a different spread and unit, a composition of kernel is generally preferred.

The only parameter of the RBF kernel is the kernel width which determines the spread of the local influence for each datapoints. It will be the kernel function used to investigate the capabilities of GPR in this document and it is defined as :

$$k(x, x') = exp(-\frac{||x - x'||_2^2}{2l^2}) \tag{12}$$

Finally if one has insights on the global evolution of the data, the function m(x) can be set accordingly. If it is not the case, the mean is usually subtracted to the data such that the prediction far from the data (equal to 0 with an RBF kernel) correspond to the mean of the data.

B. Sensitivity to outliers and gaps

GMR predicts the value of the function based on a weighted sum of the prediction of each component. Outside of the dataset, GMR will therefore typically predict the latest observed trends in the data, as the Gaussian that fits this trend will be dominant in the vicinity. However, this prediction may be affected by other mixtures and become nonsensical as one get far from the last observed data. Its prediction capability where there are gaps inside the range of the dataset really depends on the number of Gaussian components and the data. GPR associates each point with a given kernel. Far from the training samples, in the case of the Gaussian RBF kernel, the function will decay to zero relatively fast (depending on the kernel width). Thus its prediction capability is quite limited. Using large RBF kernel, other kernel types or a mean function can improve this capability.

An outlier is a point that stands apart from the majority. As Gaussian mixtures tends to maximize the likelihood, it will fit according to the spread of the data and fit a Gaussian where the density is maximal. Outliers will have little to no influence. GPR on the other hand associate a kernel to each datapoint. If the outlier is surrounded by much training data and/or if the noise level (σ_e) is set high, the contribution of this outlier to the prediction will be small. However, it will strongly affect the prediction if there is few data in its surrounding and a the noise level is set to a small value.

C. Sensitivity to sparsity

Training a model with a low number of data samples can be the source of many problems and poor results depending of the algorithm used and its parameters. The intrinsic properties of GMR makes the task particularly difficult in that regard. Computing parameters with a high confidence while the number of datapoints is low is not possible. Beside, GMR tends to favor the areas where the density of datapoints is large as it fits the Gaussian components by maximising the likelihood of the model. Sparsity in a dataset translates to low densities and many points being far from their actual neighbours. GPR is expected to behave better in such datasets. If the data is sparse a RBF kernel should have a large width in order to interpolate between the points. However, a large kernel width doesn't enable the model to fit small variations. This can be a problem if the data has regions that are sparse (needing a large kernel) and other dense regions with small high frequency variations that one wants to capture (needing small kernel width). It is possible to use non-stationary covariance for the kernel functions that can then encapsulate local variations in the density of the datapoints. This will not be presented here.

D. Sensitivity to initialization

In the first step, GMR fits a GMM to the data. This is done by expectation-maximization (E-M) of the model Likelihood. The result of these E-M steps is sensible to the initialisation of the centroids. When the hyperparameters of GPR are fixed, the model is derived from the maximum a posteriori estimator

(MAP) which has a closed form solution. Thus, it is not sensible to initialisation. In contrary, when the parameters are optimised, the optimization problem is not convex and may stop at a local minima. It is hence sensible to initialisation.

E. Computational complexity

In this section the number of datapoints is denoted M and the number of dimensions is denoted N.

For training, GPR requires the computation of the inverse of a $M \times M$ matrix (the Kernel matrix, also called the Gramm matrix). This operation has a high complexity of $\mathcal{O}(M^3)$. For testing, as GPR keeps all the datapoints for computation, the complexity remains unchanged. That is quite undesirable as testing is generally expected to be faster than training.

For GMR, the complexity relies in the Expectation-Maximization algorithm that is used to find the parameters. For training, the dominant term is $\mathcal{O}(K \cdot N^2)$ which appears in the maximization step for the computation of the covariance matrices. However this expression gives a rather poor estimation for mixture models as the E-M algorithm requires an arbitrary number of iterations to converge. Some author rather write $\mathcal{O}(F \cdot K \cdot N^2)$ with F the maximal number of iterations in the E-M step [8]. For testing, the complexity is $\mathcal{O}(K \cdot N^2)$. In contrary to GPR, GMR can handle a multidimensional output in one swipe. In the case of GPR, one has to do a pass per output dimension.

F. Application on the dataset

Finally, we compare both algorithm on a real dataset which concerns the movement of a 8 joint robotic arm. It contains 8192 records of each pivot angle (θ_i). The output is the distance from the base of the robot to its end-effector. The dataset is advertised as being highly non-linear and featuring medium noise according to the Delve repository [12]. This is a typical problem to which it is difficult to derive by hand a closed-form solution and where machine learning is used.

The algorithms are implemented on Matlab® R2019b using the built-in functions from the software as well as the *ML toolbox* developed at LASA, EPFL [11].

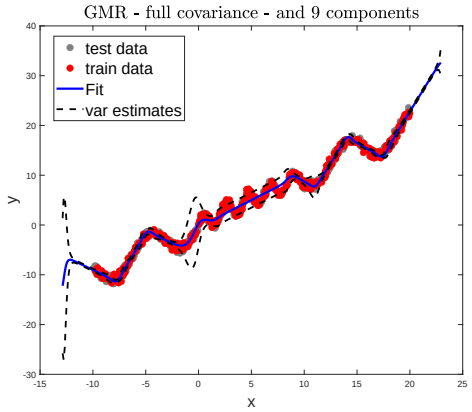
To find the number of Gaussian component needed for our dataset, we will perform crossvalidation with 10 Folds and compute the AIC (Akaike information criterion) and BIC (Bayesian Information Criterion) [9].

To analyze the quality of the regression, the Mean Square Error (MSE), the normalized mean squared error (NMSE) and the coefficient of determination (R^2) are computed on the testing set during the crossvalidation.

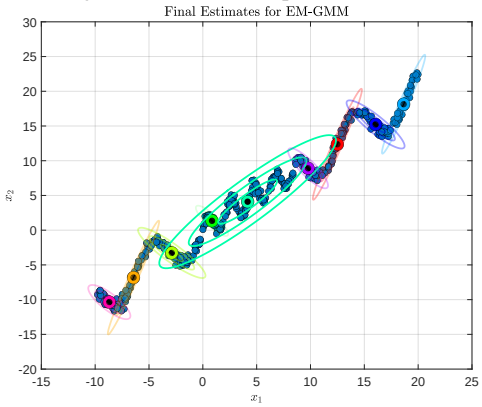
III. Results

A. Hyperparameters

As one can see by comparing figure 1 and 2, increasing the number of Gaussian components enables to fit the small variations present in the center of the non-linear curve. On figure 3, we can observe that a small kernel width with a small σ_e will better capture small variations, while a big kernel width with a big σ_e will smooth out the small variations. On the subfigure 3c, one can observe that with a small σ_e, the points at the upper left corner of the images are better fitted than on subfigure 3b since the signal is considered less noisy. Setting a small noise also reduces the predictive variance of the model as it can be seen on subfigure 3c.



(a) Regression curve with predictive variance.



(b) Locations of the Gaussian components used to predict the curve.

Fig. 1: GMR regression with 9 Gaussian components, the small variations are not encapsulated.

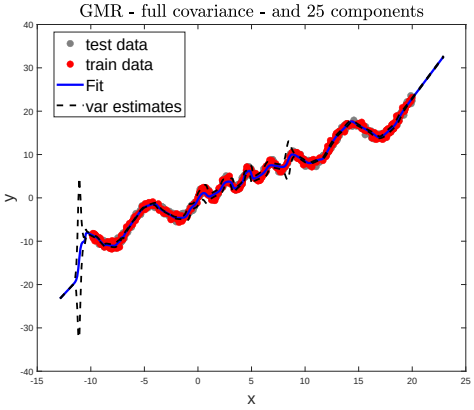
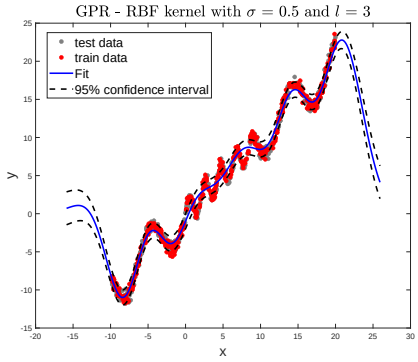
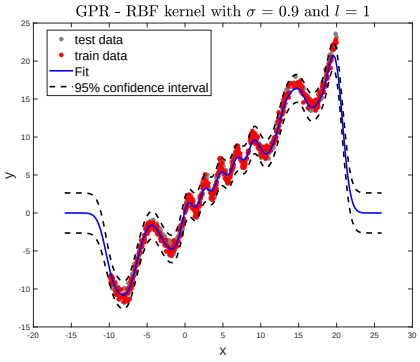


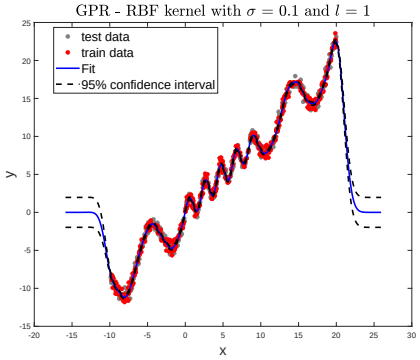
Fig. 2: GMR regression with 25 gaussian components, the small variations are encapsulated.



(a) GPR regression with a big kernel and a big noise variance.



(b) GPR regression with a small kernel and a big noise variance.



(c) GPR regression with a small kernel and a small noise variance.

Fig. 3: GPR regression with different hyperparameters.

Finally, the effect of setting a linear mean function is shown on figure 4. We see that far from the data the prediction doesn't fall to 0 but goes to this mean function. In the next sections, this function is set to zero to simplify the discussion.

B. Sensitivity to gaps

The test consists of the removal of all data for $x \in [0; 10]$ in the toy dataset. Looking first at GPR, it's evident that the width of the kernel plays an essential role. All points close to the gap will add with a certain weight defined by α_i. If the data possess some trend close to the border, GPR will typically follow this trend on a given range determined by the kernel width as it can be seen on figure 5. Since the dataset is fairly dense, GPR does not decay after the last point. However, we do see that in the scale of the width, the function starts to drop.

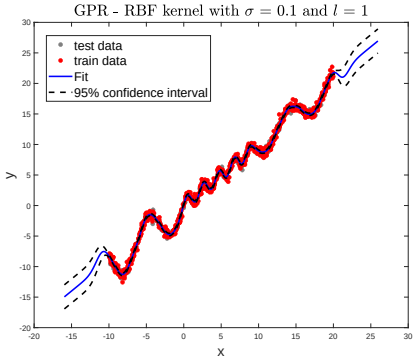


Fig. 4: GPR with a linear mean function.

The exact same analysis can be done with the right hand side of the curve, where a plateau is observed near the data. If the kernel width is too large, the effect of many kernel will be summed and the prediction will not match the actual curve.

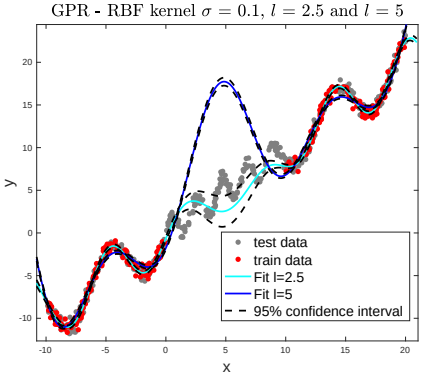


Fig. 5: GPR behaviour upon a gap in the data, depending on the kernel width.

In the case of GMR, the prediction strongly depends on the number of Gaussian, where they are placed and their relative importance. As one can see on figure 6, the Gaussian mixture on the right-hand side of the gap is responsible for more points and has a variance that is more elongated towards the gap that the one on the left. Therefore, it influences more the prediction in the gap which gives a bad prediction in this case.

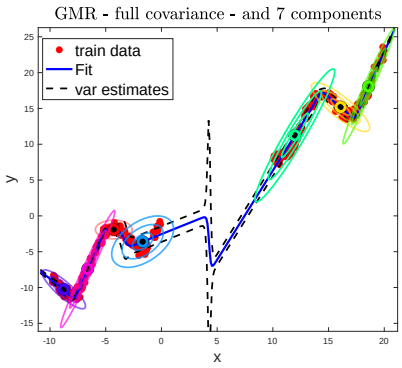


Fig. 6: GMR regression with estimates, behaviour over a gap.

C. Sensitivity to sparsity

In this section, only 7% of the training data was kept, summing up to a total of only 35 training samples on the same toy dataset. GPR behaves particularly well in this situation. It does fail to grasp the small ripples in the middle of the curve but solely due to the clear lack of points (fig. 7). The kernel width was adjusted to $l = 2$ to account for the longer mean distance between each point.

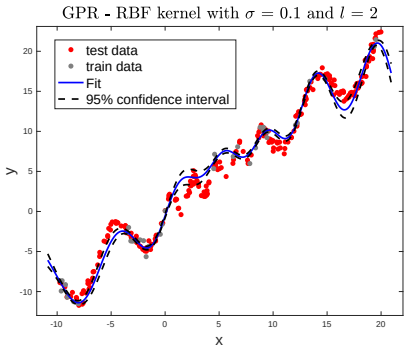


Fig. 7: Gaussian Process Regression with only 7% of the original training data.

Sparsity is a harder task for GMR which needs a sufficient amount of datapoints to properly estimate the distribution of the dataset. Performing cross-validation on GMR with K ranging from 1 to 15, we find that, $K = 10$ gives the best BIC score. However, while the curve fits mostly the training data, the curve is highly non linear and the distribution of the data is not well estimated. Reducing the number of components to 5 leads to a somewhat acceptable result after many trials (fig. 8). Since the data is very sparse, the position of the Gaussian components greatly depends on the initialisation and most often they lead to bad results. Reducing to $K = 4$ does not yield better results and repeating the GMR algorithm over 20 times, picking different samples each time, did not yield anything better. The resulting regressive model either have strong discontinuities or simply fail to describe the data.

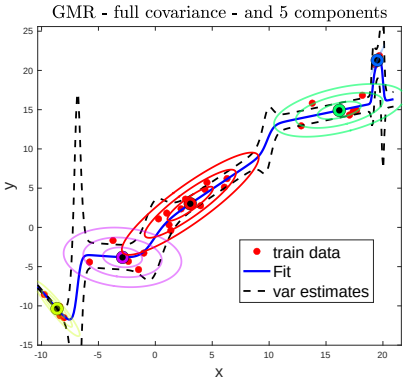


Fig. 8: Resulting mixtures and regression, the model misses most of the subtleties of the data in the center region with sparse data.

D. Application on the dataset - Introduction

Upon discovering a dataset, one can first take a look at the standard deviation of each feature and the output. As they are all recordings from joints of a robot, they all have a standard deviation around 0.9. The output has a standard deviation of 0.26. It gives a hint of the scales we're dealing with. From this, we can estimate a range for the kernel width. If each dimension had widely different distributions, the kernel width would have been unsuitable for some of them. In this case, normalizing all dimensions by their standard deviation would be a requirement.

Due to the limited possibilities at visualizing the data distribution due to the high number of dimensions (8 input, 1 output) and the large number of entries (8192), there's no proper way to start off without going straight to a cross-validation test for both algorithm. One could try to project every dimension against each other and get a glimpse of the data arrangement but it may not be helpful at all. Unfortunately, the noise in the dataset is not quantified. Data was solely advertised as being contaminated with medium noise. Noise will affect the regression metrics, a good regression may thus not be described by a curve that gives $R^2 \approx 1$ but rather $R^2 < 1$, as one does not want to fit noise.

E. Application on the dataset - GMR

As we expect the output to be nonlinear, we choose full covariance matrices. Being clueless about the shape of the output with respect to the input dimensions, we start straight with cross-validation to locate a region where R^2 is high and MSE, BIC, AIC are low altogether.

Setting the training-testing ratio at 50%, we have about 4000 points for training so the model should have at most 400 or so parameters, if we follow the rule of the thumb given in equation 11. If this is not satisfied, the parameters will still be computed but with low confidence. Starting off with a first cross validation pass, k ranging from 2 to 20, with ten folds for each number of components, we obtain fig. 9.

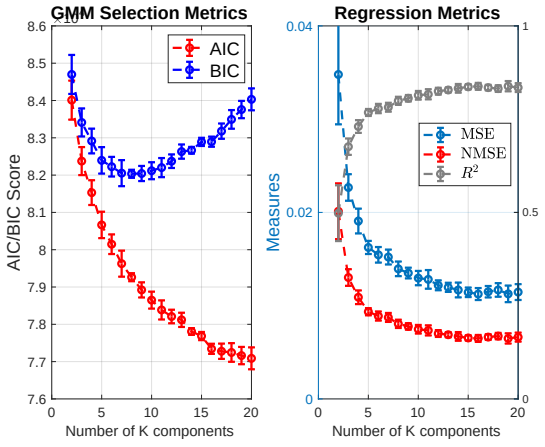


Fig. 9: Cross-validation, 2 to 20 components.

As we can see, for low values of K, the curves are evolving fast, this region is not of interest as the model is grasping more specificities of the data as K grows. While the AIC curve is not particularly helpful, the BIC curve tends to indicate a minimum at $K = 7$. Beside, we notice two significant elbows in the MSE curve: one at $K = 5$ and another at $K = 7$. The metrics tends to indicate that $K = 7$ is a good comprise between number of components and the performance of the model. The coefficient of determination remains remarkably low unfortunately.

One may suspect that this is due to the fairly low ratio used for training. As we noticed earlier, GMR tends to have some difficulties when points become sparse. However, since R^2 has a very small standard deviation, it informs us that R^2 scored consistently the same value of 0.8 for $K = 7$ as the quartiles are nearly in the point itself. It tends to show that this is not a problem of having too few training points. Increasing the train-test ratio to 75% yielded identical results, GMR did not perform any better. However, this does not mean the model performs badly as the data has medium noise. It would be quite interesting to use a simulator to see how the regression actually performs.

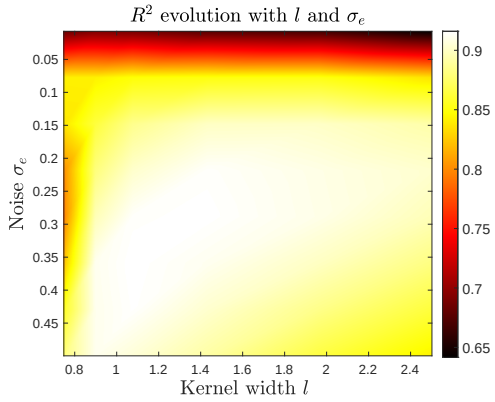
F. Application on the dataset - GPR

Unlike GMR, we have some elementary clues for one of the hyperparameter thanks to the standard deviation in each dimension. Of course, this just gives an estimate. Running the algorithm with the following kernels widths: $\{0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25\}$ and the various noise $\sigma_e = \{0.01, 0.05, 0.1, 0.2, 0.35, 0.5\}$, results in figure 10.

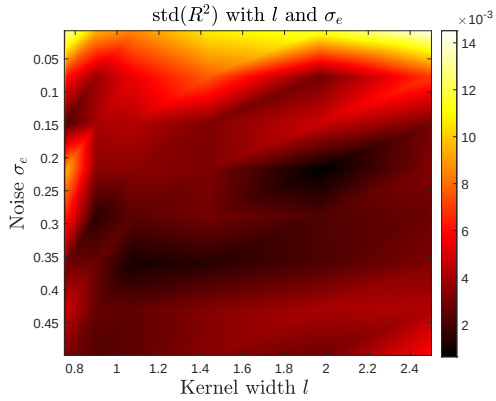
For GPR, two hyper parameters impose to work with colormaps. We observe a maximum region in the range of $\sigma_e \in [0.2; 0.4]$ and $l \in [1; 2]$. We see that the performance of the algorithm based on the R^2 metric is much better than the one obtained with GMR. In order to assess if this is due to GPR overfitting the data. We show the standard deviation of R^2 on figure 10b and choose a region where it is small while R^2 is close to 1 in order not to have an overfitted model : $l = 1.28$ and $\sigma_e = 0.35$ giving $R^2 = 0.9122$ with a standard deviation of 0.0014.

The MSE values are typically of 0.0059 which is half of the MSE obtained with GMR.

Beside doing a grid-search, many softwares can optimize the kernel parameters themselves. Optimizing the kernel width l and σ_e with Matlab® built-in functions with 200 optimization passes results in $l = 1.74$, $\sigma_e = 0.1936$ with the associated metrics: $R^2 = 0.917$ (standard deviation of 0.0018 evaluated by a cross-validation afterward) and $MSE = 0.006$. Those results are totally compatible with the colormap observed on fig. 10. The total computational time is about as long, as 200 passes is close to $F_f \cdot N_{widths} \cdot N_{noises}$. Both techniques remain complementary, as optimization does not provide the standard deviation of the metrics, which can be useful to determine the quality of the model.



(a) Cross-validation considering various kernel width and noise.



(b) Cross-validation standard deviation across the map.

Fig. 10: Cross-validation with R^2 as a metric.

IV. Discussion

A. Regression performance

In all the conducted experiments, GPR always provided extremely smooth curves. One of the source of this phenomenon is the usage of the RBF kernel, which is infinitely differentiable. On the other hand, due to its nature GMR will tend to offer less linear curves. GPR, depending of the kernel, could struggle to describe a discontinuity in the data while GMR could handle it.

B. Behaviour on region with no data

GPR with a RBF kernel is somewhat influenced by the arrangements of the points near a region without training samples but one cannot really talk about predicting abilities as the function will goes to zero soon after. On the other hand, GMR prediction can be particularly valuable on the outer edge of the dataset. It may not be reliable in gaps of the training data as discussed in section III-B. One should keep in mind that GPR does feature a mean function which can potentially describe the data even in region lacking samples. This feature was not investigated here.

C. Sparse data

GMR clearly under-performed when the training samples were sparse. It struggles to estimate correctly the distribution

of the dataset. GPR behaved extremely nice on the other hand and still gave good results with less datapoints. This is probably one of the key strength of GPR compared to GPR.

D. Application to real dataset

GPR definitely performed better than GMR on the *pumadyn-8nm* dataset. But again, it would be interesting to simulate the resulting model. However, with an eight-dimension input, we can question the usage of the RBF kernel which relies on the L2-norm. This norm usually become less meaningful as the number of dimensions increases [6]. For the example used, the L2 norm present in the kernel may have lead the model to give poorer performance. We also observed that the total number of the model parameters with a train test ratio of 0.5 gives a total of 4096 points with 8 dimensions + 2 hyperparameters to memorize for GPR. GMR requires the storage of only 329 values (equ. 10). This shows that GPR is more memory demanding.

V. Conclusion

Both algorithms having vastly different properties backs the fact that they're rather alternatives to each other rather than competing algorithm. Either should be picked depending of the task (dimensions, size, noise, sparsity, ...) and the actual needs imposed by the task that should be performed. If the goal is to get the most precise regressive curve with respect to the data, GPR would be a good choice. However, if one wants to teach a robot movements, GMR would be more interesting due to the intrinsic predicting abilities and the smoothing it can provides to noisy data. Lastly, at testing GMR is significantly faster than GPR for large datasets so if time is critical, GPR could be excluded. There exists some sparse method for GPR that try to reduce the number of parameters by selecting only a subset of representative datapoints so that the algorithm is faster at retrieval time [10]. We can also note that, in contrary to GPR which assumes constant noise, GMR can capture the local variance of the estimate which makes it interesting to use with heteroskedastic data.

References

[1] Sung, Hsi Guang. Gaussian mixture regression and classification. Diss. Rice University, 2004.

[2] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*, the MIT Press, 2006, www.GaussianProcess.org/gpml

[3] Williams, Christopher KI. "Prediction with Gaussian processes: From linear regression to linear prediction and beyond." Learning in graphical models. Springer, Dordrecht, 1998. 599-621.

[4] A. Billard and A. Polydoros. "Non-linear regression part III". Slides for the course *Advanced Machine Learning*, EPFL. Rev. 6 May 2020.

[5] Unknown author. *Advanced Machine Learning Algorithms "A comprehensive study"*. 2020 Mini-project guidelines, EPFL, Advanced Machine Learning course.

[6] Aggarwal C.C., Hinneburg A., Keim D.A. (2001) On the Surprising Behavior of Distance Metrics in High Dimensional Space. In: Van den Bussche J., Vianu V. (eds) Database Theory — ICDT 2001. ICDT 2001. Lecture Notes in Computer Science, vol 1973. Springer, Berlin, Heidelberg

[7] Duvenaud, D. (2014). Automatic model construction with Gaussian processes (Doctoral thesis). https://doi.org/10.17863/CAM.14087

[8] A. Billard and al. *Non-linear regression techniques Part - III*. Advanced Machine Learning course, EPFL, Lausanne. Slides for the 2019-2020 academic year.

[9] Burnham, K. P., Anderson, D. R. (2004). Multimodel Inference: Understanding AIC and BIC in Model Selection. Sociological Methods Research, 33(2), 261–304. https://doi.org/10.1177/0049124104268644

[10] J. Schreiter, P. Englert, D. Nguyen-Tuong and M. Toussaint, "Sparse Gaussian process regression for compliant, real-time robot control," 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, 2015, pp. 2586-2591, doi: 10.1109/ICRA.2015.7139547.

[11] Billard, A., Figueroa N., (2020) *ML toolbox https://github.com/epfl-lasa/ML_toolbox*

[12] Dataset used : *https://www.cs.toronto.edu/ delve/data/pumadyn/desc.html*

The real dataset was obtained from the Delve Repository. PUMA DYNamics family of datasets. From which, we focused on *pumadyn-8nm*. Some authors also refer to this dataset as *kin8nm*.