1 Algorithm details

Let the training dataset be represented by $\{(x_i, y_i)\}_{i=1}^{N}$, where x_i are the inputs and y_i are the outputs of the function to be approximated, and both can be multidimensional. Let K be the number of Gaussian kernels to fit to the data. A single datapoint in the training dataset can be represented by $d_i = \{(x_i, y_i)\}$. A mixture model of these K components can be defined by a probability density function as follows:

$$p(d_i) = \sum_{k=1}^{K} p(k)p(d_i|k) \tag{1}$$

Where for Gaussians of dimensionality D:

$$\begin{aligned} p(k) &= w_k \qquad \text{such that} \qquad 1 = \sum_{k=1}^{K} w_k \\ p(d_i|k) &= \mathcal{N}(d_i; \mu_k, \Sigma_k) \\ &= \frac{1}{\sqrt{(2\pi)^D |\Sigma_k|}} e^{-\frac{1}{2}\left((d_i-\mu_k)^T \Sigma_k^{-1}(d_i-\mu_k)\right)} \end{aligned} \tag{2}$$

$\{w_k, \mu_k, \Sigma_k\}$ represent the parameters of the k th Gaussian, where w_k represents the prior (or weight), μ_k represents the mean, and Σ_k represents the covariance. The iterative Expectation-Maximization (EM) algorithm is used to optimize these parameters by finding the maximum likelihood estimates (MLE). After the EM algorithm has converged, the Gaussians have been fitted to the data and training is complete. Regression using the parameters is performed during prediction.
The operation of the GMM-GMR algorithm [1] is detailed as follows:

Training - Fitting GMMs using the EM algorithm

1. Initialize the parameters. Means μ_k are initialized using k-Means clustering. Covariances Σ_k are initialized as identity matrices. Weights w_k are uniformly initialized to $\frac{1}{K}$.
2. **Expectation** step of EM - Calculate the likelihood that each Gaussian generated the data
   (a) Calculate the responsibility $\gamma_{k,i}$ of each Gaussian k for each datapoint d_i:

$$\gamma_{k,i} = \frac{w_k \mathcal{N}(d_i|\mu_k, \Sigma_k)}{\sum_{k=1}^{K} w_k \mathcal{N}(d_i|\mu_k, \Sigma_k)} \tag{3}$$

   (b) Calculate the responsibility Γ_k of each Gaussian k for all the data:

$$\Gamma_k = \sum_{i=1}^{N} \gamma_{k,i} \tag{4}$$

3. **Maximization** step of EM - Update model parameters to increase the likelihood calculated from the E-step. Update the weights w_k, means μ_k, and covariances Σ_k for each Gaussian:

$$w_k = \frac{\Gamma_k}{N}, \qquad \mu_k = \frac{\sum_{i=1}^{N} \gamma_{k,i} d_i}{\Gamma_k}, \qquad \Sigma_k = \frac{\sum_{i=1}^{N} \gamma_{k,i}\left((d_i - \mu_k)^T \Sigma_k^{-1}(d_i - \mu_k)\right)}{\Gamma_k} \tag{5}$$

4. Repeat steps 2 and 3 until the change in likelihood of the model generating the observed data is below a threshold ($1e^{-4}$ in the code implementation), or when the maximum number of iterations has been reached (200 in the code implementation).

Prediction - Regression using the K Gaussians' parameters

1. For each Gaussian component fitted, separate the input and output means μ_k and covariances Σ_k.

$$\mu_k = \{\mu_{x,k}, \mu_{y,k}\}, \qquad \Sigma_k = \begin{pmatrix} \Sigma_{xx,k} & \Sigma_{xy,k} \\ \Sigma_{yx,k} & \Sigma_{yy,k} \end{pmatrix} \tag{6}$$

2. For each Gaussian k,

   (a) The conditional expectation (i.e. prediction) of y_k given x_q is

$$\hat{y}_k = \mu_{y,k} + \Sigma_{yx,k}\left(\Sigma_{xx,k}\right)^{-1}(x_q - \mu_{x,k}) \tag{7}$$

   (b) The estimated conditional covariance of y_k given x_q is

$$\hat{\Sigma}_{yy,k} = \Sigma_{yy,k} - \Sigma_{yx,k}\left(\Sigma_{xx,k}\right)^{-1}\Sigma_{xy,k} \tag{8}$$

3. Weight each Gaussian kernel's prediction based on the probability that that Gaussian component explains the data in the input space. Calculate the weight for each kernel β_k:

$$\beta_k = \frac{p(x_q|k}{\sum_{j=1}^{K} p(x_q|j})} \tag{9}$$

4. Weight the prediction and the prediction covariance by β_k:

$$\hat{y}_q = \sum_{k=1}^{K} \beta_k \hat{y}_k, \qquad \hat{\Sigma}_{yy} = \sum_{k=1}^{K} \beta_k^2 \hat{\Sigma}_{yy,k} \tag{10}$$

2 Testing on simple datasets

To verify the functionality of the implemented GMM-GMR algorithm, the algorithm was trained and tested on simple, low-dimensional (1 input, 1 output) toy datasets with known groundtruth. The toy datasets were generated from (a) a linear function with positive slope, (b) a linear function with negative slope, and (c) a sine function. Gaussian noise was added to all toy datasets. The results are shown in Figure 1.

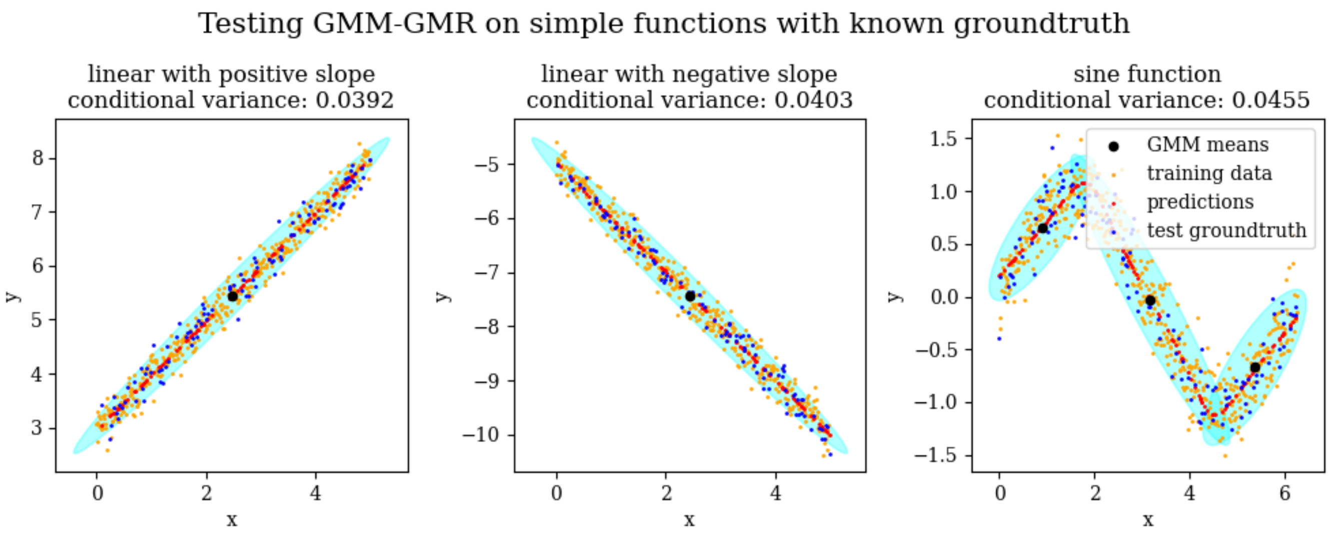

Figure 1: Demonstration of algorithm implementation working on simple datasets

GMM-GMR successfully recovers the trends in the toy datasets. Clusters fitted by GMM are shown as cyan ellipses on the plots. The number of clusters to fit, K, however, needs to be manually selected. For these toy datasets, it is easy to determine K by inspection of the plots showing the relationship between all the dimensions - for instance, the sine wave in the right-most plot of Figure 1 could be modelled by distinct clusters for each of the 3 sections $x = [0, \frac{\pi}{2})$, $x = [\frac{\pi}{2}, \frac{3\pi}{2})$, $x = [\frac{3\pi}{2}, 2\pi]$. For the robot dataset however, this is non-trivial due to the large number of total input and output dimensions, and thus resulting complex co-covariances. Section 4 elaborates on the process for K selection.

3 Dataset pre-processing

To keep training within a tractable scale for this assignment, the dataset was downsampled from the original 95809 samples to 9581 samples by sampling every 10th datapoint in the dataset. Testing data was extracted from the downsampled data to be held out during training (unseen) and used for model evaluation. This extraction was done by further sampling every 5th datapoint in the downsampled dataset. The final train-test split consisted of 7664 training datapoints and 1917 training datapoints (an approximate 80/20 split).

Before training on the dataset (whether using my implementation or sklearn), the data was randomly shuffled row-wise (using numpy.random.shuffle) to reduce biases in training from time-sequential data.

With multidimensional data, differences in scale of the data along the different dimensions may bias learning towards the dimension whose scale dominates, and thus feature scaling needs to be carried out. *Standardization* was applied along each dimension of the data. Standardization scales values such that they are centered around a mean of 0 with a unit standard deviation, as detailed in equation 11.

$$X' = \frac{X - \mu}{\sigma} \tag{11}$$

4 Model selection (choosing an appropriate K)

The Bayesian Information Criterion (BIC) is the criterion chosen for evaluating the fitted GMM. BIC for GMMs is defined as

$$\text{BIC} = K \ln(N) - 2\ln(\hat{L}) = N \ln(\frac{RSS}{N}) + K \ln(N)$$

where

- K is the number of Gaussian components to fit
- N is the number of training data samples
- $\hat{L}$ is the MLE of the entire model M ($\hat{L} = p(x|\hat{\theta}, M)$ where $\hat{\theta}$ are the model parameters $\{w_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ and x is the training data
- RSS is the residual sum of squares $RSS = \sum_{i=1}^{N}(\hat{y}_i - y_i)^2$

5 Model evaluation metrics

The metrics used to evaluate trained models are mean squared error (MSE) and R^2 score defined as:

$$\text{MSE} = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N} \qquad R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{12}$$

where N is the number of data points for which prediction is performed, y_i is the ground truth, $\hat{y}_i$ is the model's prediction, and $\bar{y}$ is the mean of the ground truth outputs.

MSE assesses the quality of a predictor, quantifying the variance and bias of the model. The lower the MSE, the more accurate the model's predictions are. R^2 score is also known as the Coefficient of determination or Goodness of fit. From the equation above, R^2 score compares how accurate the model's predictions are compared to the null hypothesis (i.e. the mean, as seen in the denominator in the R^2 expression).

References

[1] Calinon, S., Guenter, F. & Billard, A. (2007). On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *37*(2), 286–298.