Prac W12 - Gaussian Processes

COMP4702/COMP7703 - Machine Learning

Aims:

- To complement lecture material in understanding Gaussian processes (GPs) for regression.
- To gain experience with simulating and implementing Gaussian process regression in software.
- To produce some assessable work for this subject.

Gaussian process regression:

Note: This Prac does not refer directly to the textbook, so the notation will be slightly different to that seen in lectures. However, dealing with different notation between resources is a fact of life in Machine Learning so it is good experience for the course.

- Read §2.2 of Gaussian Processes for Machine Learning, freely available here.

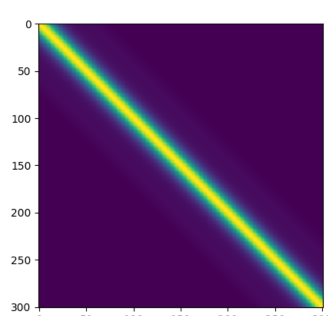
(**Q1**) The squared exponential covariance function is given by

$$k(\mathbf{x}_p, \mathbf{x}_q) = \exp\Big(-\tfrac{1}{2\ell^2}\|\mathbf{x}_p - \mathbf{x}_q\|_2^2 \Big),$$

where $\|\cdot\|_2$ denotes the Euclidean norm and ℓ is a hyperparameter called length-scale.

Write a function `K`, which accepts three inputs. The first and second inputs should be an $n \times d$ array X and an $m \times d$ array X_* respectively. The third input should be the float ℓ. The function should return an $n \times m$ array with pq^{th} element equal to k evaluated at the pth row of X and the qth row of X_*.

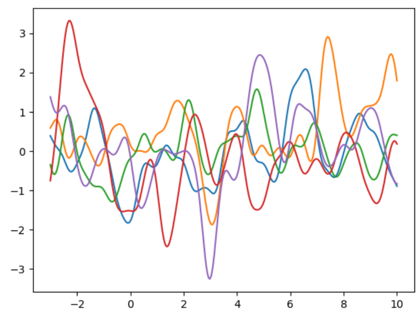
Hint: A visualisation of `K` as an image when both input arrays are the data in `test_inputs` and $\ell = 1$ is shown below.



(**Q2**) Write a function `sample_prior` which accepts three inputs. The first input should be an $n \times d$ array X. The second input should be the float ℓ. The third input N should be an integer representing the number of desired samples. The function should return an $N \times n$ array representing N samples from the GP prior with mean $\mathbf{0}$ and squared exponential covariance function with length-scale ℓ.

(**Q3**) Test the functionality of `sample_prior` by plotting 5 samples from the prior. Use the `test_inputs` data for X, $\ell = 1$ and $N = 5$. The samples should look different each time the plot is generated - why?

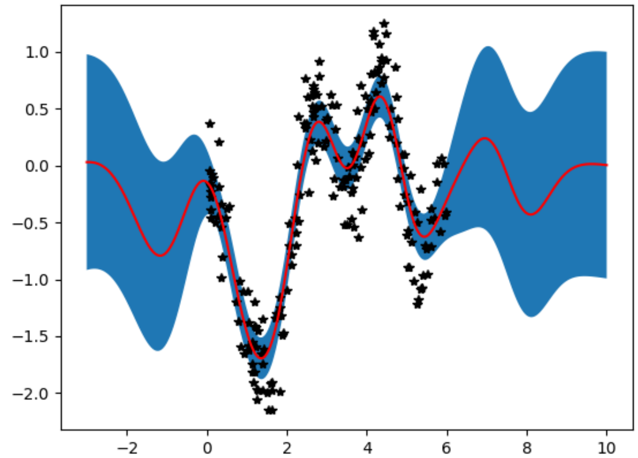
Hint: Your plot should look something like the plot below, but should have appropriate axis labels.



(**Q4**) Write a function `predictive_mean` implementing equation 2.23 of [1]. The function should take 5 inputs: The $n \times d$ array of training input data X, the $n \times 1$ array of training target data $\mathbf{y}$, the $m \times d$ array of prediction input data X_*, the float ℓ and the float representing the noise parameter σ_n. The function should return an $m \times 1$ array representing the mean of the posterior predictive distribution at X_*.

(**Q5**) Write a function `predictive_cov` implementing equation 2.24 of [1]. The function should take 5 inputs: The $n \times d$ array of training input data X, the $n \times 1$ array of training target data $\mathbf{y}$, the $m \times d$ array of prediction input data X_*, the float ℓ and the float representing the noise parameter σ_n. The function should return an $m \times m$ array representing the covariance of the posterior predictive distribution at X_*.

Hint: The plot below shows the predictive posterior mean and predictive posterior standard deviation (square root of diagonal entries of the covariance matrix) on top of the training data, when X is `train_inputs`, $\mathbf{y}$ is `train_outputs`, X_* is `test_inputs`, $\ell = 1$ and $\sigma_n = 1$.



(**Q6**) Test the functionality of `predictive_mean` and `predictive_cov` by plotting 5 samples from the predictive posterior with the training data. Use the `train_inputs` data for X, `train_outputs` data for $\mathbf{y}$, `test_inputs` data for X_*, $\ell = 1$, $\sigma_n = 1$ and generate $N = 5$ samples. The samples should look different each time the plot is generated - why?

Hint: Your plot should look something like the plot below, but with labelled axes.

Datasets:

All of the data in this practical is taken from the SPGP .zip folder on this website `http://www.gatsby.ucl.ac.uk/~snelson/`

Useful Python Commands:

`scipy.spatial.distance.cdist`, `numpy.random.multivariate_normal`, `matplotlib.pyplot.fill_between`

Useful Matlab Commands:

`pdist2`, `mvnrnd`, `fill`

Additional Resources:

[1] Gaussian Processes for Machine Learning, freely available here.

[2] You can swap out the kernel from **Q1** for different kernels. A gentle introduction to other kernels is here.