Could run a couple of evaluations to generate an initial sample set.

Think of a good metric to decide when to add samples to your model.

PsuedoCode GP CSWK 1 PART 2

**Generic model**

**Creating the Bayesian\_Gaussian Class (it is a Gaussian process class model,** but the assignment name in the code is Bayesian\_Gaussian**)**

Initialised methods (default methods):

You can decide what kernel function to apply and noise input.

The kernel function allows us to measure similarity between our input in order to decipher if there is correlation.

The noise function captures uncertainty, increasing the generality of the model and can help prevent overfitting (to an extent).

**Fit method:**

X\_TRAIN:

Forms a (n,2) array: where n is the number of samples, and there are 2 features within this model since it is a 2d problem.

Y\_train:

Forms a (n,1) array: where n is the number of samples.

K:

The K parameter produces the kernel matrix with some noise added to it.

The kernel function models the **covariance** of the output and the input by seeing how similar the different samples of an input is (i.e. measure the variance of the sample data), the covariance is modelled assuming that if the sample has a high variance then there should be a low expected co-variance (the outputs would be far apart and you wouldn’t be able to infer the directional relationship of outputs with inputs).

The kernel also helps with smoothness.

A white paper with black text

Description automatically generated

**Kernel function:**

In our model we don’t have a variance parameter but we have a length scale=1.(so identical to black screenshot)

**A screenshot of a math problem

Description automatically generated**

**A black background with white text

Description automatically generated**

In the case of our model:

Kernel matrix dimensions will be a square matrix (nxn)where n is the number of our samples

Minimum of 2 points, needed for covariance so feed it 2 points, points will increase since we are iteratively training the model.

The K matrix is imperative in training our model as it enables us to model for covariance which is necessary for our GP regression.

K\_inv:

The inverse of the K matrix is evaluated will also be an nxn matrix.

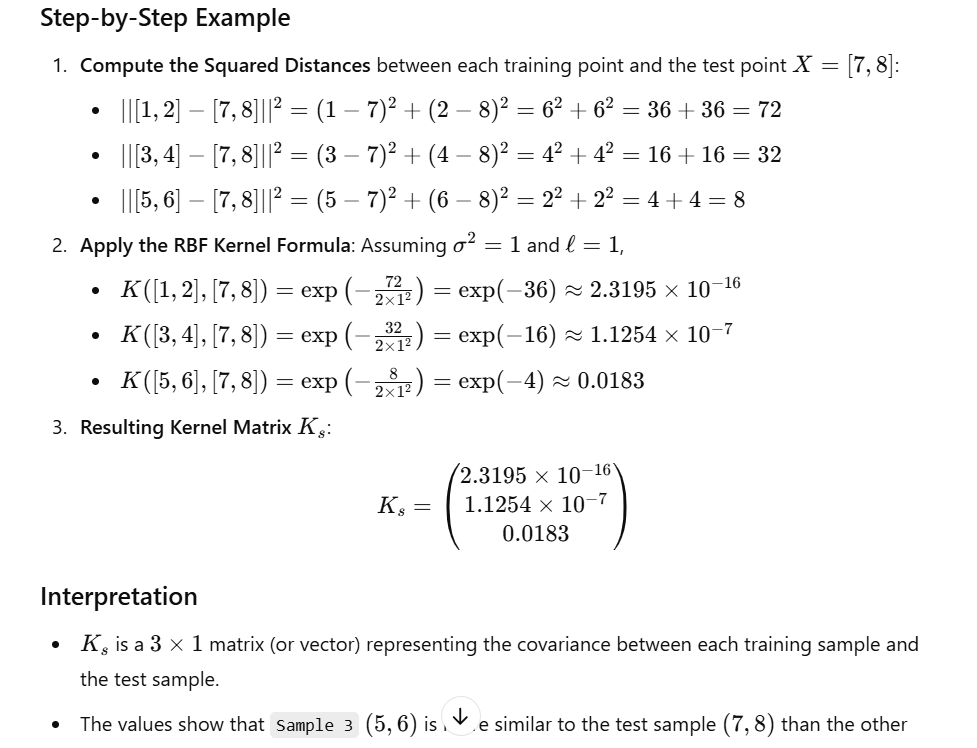
**Predict method:**

Kernel\_predict:

This method is designed to model covariance between the sample data and the new test point

A screenshot of a math test

Description automatically generated



A screenshot of a test

Description automatically generated

Produces an (nx1) vector where n is the number of samples.

Uses a modified kernel function that is able to take a single sample

K\_ss:

Covariance of a single test point therefore we will get a (1x1) array (which will be 1), we also add some noise (assuming test point is singular).

Mu\_s:

Our predicted mean and covariance determined via A screenshot of a math equation

Description automatically generated

For the test point, both will be scalars.

For a sample of test points it will be a square matrix.

**Expected improvement:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a math test

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A close-up of a white background

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated