4F13 Machine Learning Coursework 1 : Gaussian Processes

Submitted By: Riashat Islam (303190617) **Taught By: Professor Zoubin Ghahramani**

Question a

The dataset is trained with a Gaussian Process, having a squared exponential covariance matrix with hyperparameters hyp.cov=[-1 0]. The GPML toolbox uses these values in log scales. Therefore, actual and optimized values of length scale and variances are given below:

Initial hyperparameters:
Length_scale = exp(-1) = 0.3679
Signal Variance = exp(0) = 1

Optimized hyperparameters:
Length_scale = 0.1282 Signal Variance = 0.8046

The predictive distribution using the above hyperparameter initializations is given below. The initialized hyperparameters hyp.cov=[-1 0] for the squared exponential covariance function is too high to fit the data; therefore, the optimized hyperparamter values are reduced to make covSEiso suitable. The following code were used (without any linear mean):
covfunc = @covSEiso; hyp.cov = [-1; 0]; hyp.lik = 0; nlml = gp(hyp, @infExact, [], covfunc, likfunc, x, y);

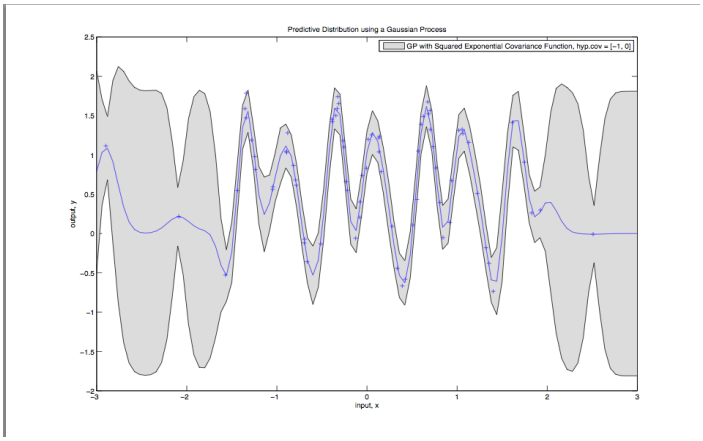


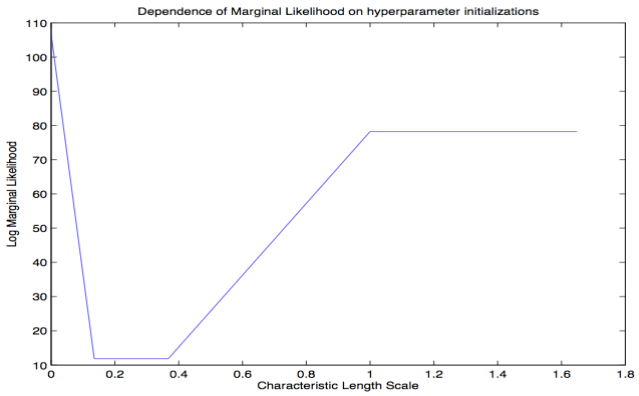
Figure 1: GP fit of the data with 95% predictive error bars

The plot below is for the optimized mean and variance at the sample test locations, with the plot for the mean function plus/minus two standard deviations corresponding to the 95% confidence interval. The error bars (confidence interval) shows that in regions where there are less test points, the model fit is more uncertain about the predictions in regions with fewer test points.

In regions of more points, the predictive distribution has reasonable confidence, with a certain level of uncertainty. The model is quite different from the data generating process in regions of few test points, whereas there is a much better fit in regions of more test data.

Question b

In this section, we train the Gaussian process – meaning, we do "model selection" and find the optimum parameters for the squared exponential covariance function to find marginal likelihood local optimum. The hyperparameters are the variance and the length scale. The length scale can be interpreted as how far we need to move in the input space for the function values to become uncorrelated. Here, we tried a range of hyperparameter initializations to see convergence of hyperparameters to a local optimum. For each initialization, we compute the log marginal likelihood, add test inputs to compute the mean and variance at test locations, and estimate the generalization error representing them with 95% predictive error bars.
The table below shows the range of hyperpameter initializations. The plot on the right shows the dependence of marginal log likelihood on length initializations. Since the marginal likelihood term includes both data fit and complexity term, and we are optimizing hyperparameter values over the entire term – a very high NLML suggests that the model is less certain about data points being generated by this model (more uncertainty in data). A very low NLML shows that the data is more likely to be generated by this model, with a good model fit (less uncertainty in GP fit). This is also illustrated by the figures below:



L init	L opt	Variance init	Variance opt	NLML
0.3679	0.1282	1	0.8046	11.899
4.54e-5	4.54e-5	1	0.4991	106.3493
0.1353	0.1282	1	0.8046	11.899
1	8.3488	1	0.4884	78.2203
1.6487	7.1644	54.5982	0.4562	78.2217

Here, we do not show the initial parameterizations of the standard deviation of the noise term. Figure 2 shows that when length scale and variances are initialized to reasonable values, the NLML is not too high or low. Figure 3 shows that with reasonable initializations (that are also close to optimized values as shown in row 1 and 3 in table above), the model fit is better, with a lower negative marginal likelihood. Figures 4 and 5 shows that the model is a poor fit (model quite different from the generating process) indicated by the shaded regions of the graph (high uncertainty in predictions). Row 2 and 4 corresponding to these graphs shows that if the length scale initialization is very small or very high, then the negative marginal likelihood is very high (indicating a very low confidence), with the model generating the data is poor (too simple). As shown in table above and the figures below, notice how the optimized hyperparameter values are significantly different and dependent on initializations of parameters. The following range of length scale and variance values were used for the for loop:
c_l = [-10, -2, -1, 0, 0.5, 1, 3, 10]; c_sf = [0, 0.5, 1, 2, 10, -10, -2, -1]; covfunc = @covSEiso; hyp.cov = [c_l(i); c_l(j)]; hyp.lik = 10;
and test points: z = linspace(-3, 3, 101)';

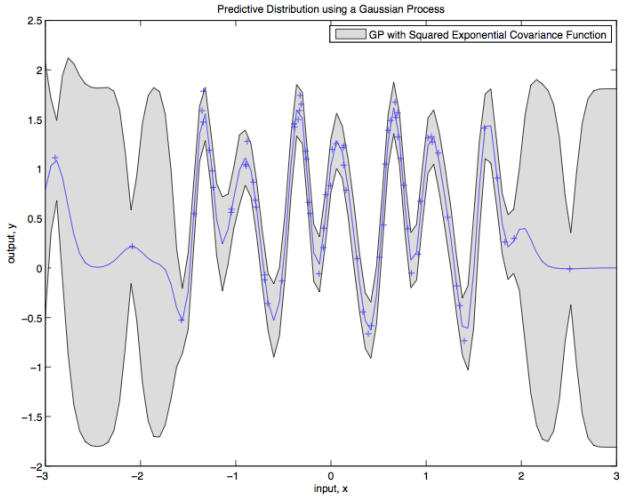


Figure 3: Corresponding to Row 1 of table above (L_opt = 0.1282, Var_opt=0.8046) and NLML = 11.899

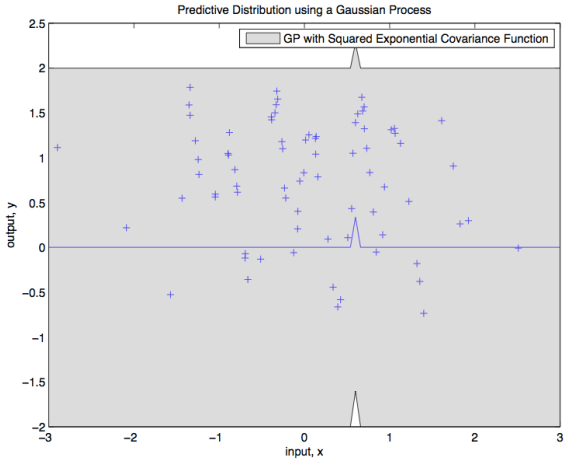


Figure 4: Corresponding to Row 2 of table above (L_opt = 4.54e-5, Var_opt=0.4991) and NLML = 106.35

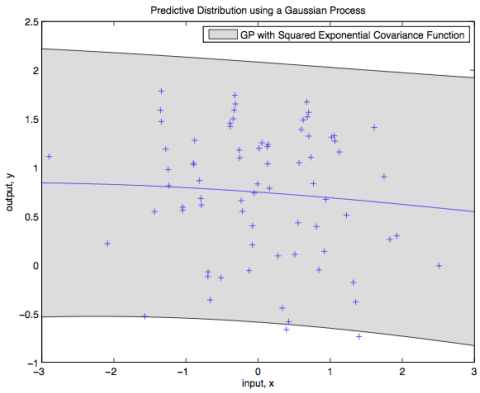


Figure 5: Corresponding to Row 4 of table above (L_opt = 8.3488, Var_opt=0.4884) and NLML = 78.2203

Explanation:

It can be concluded that initalizations of hyp.cov=[-1 0] (ie, L_init=0.3679)or hyp.cov=[-2 0] (ie, L_init=0.1353) are reasonable hyperparameter initial values, which makes the NLML to not support the model generating the data. With growing length scales, the model gets less complex, but leads to poor ability to explain the data, as shown in Fig 4 and 5.

Figure 3 shows a reasonably better GP fit to the data compared to figures 4 and 5. With a certain small range of length scale initializations, the nlml supports the data fit, with the small values (as shown in table above).

Figures 4 and 5 shows poor fit to the data with uncertainty, having very large nlml values, showing that local optimum is not achieved.

Question c

Similar to previous sections, we tried a range of values for the hyperparameters in the periodic covariance function. Below are the parameters that model the data well, with the corresponding fit as shown in figure 5. Code snippets as:

```
covfunc = @covPeriodic;   ell = 2;   np = 5;      sf = 2;   hyp.cov = log([ell; np; sf]);   hyp.lik = log(0.1);
nlml = gp(hyp, @infExact, [], covfunc, likfunc, x, y);      z = linspace(-3, 3, 101)';
```

Length_opt	Period_opt	Variance_opt	NLML
L_opt = exp(-0.7159) = 0.4888	P_opt = exp(0.6919) = 1.9975	V_opt = exp(2*0.2245) = 1.2517	-16.4378
L_opt = exp(-0.00427) = 1.0043	P_opt = exp(-0.0011) = 0.9989	V_opt = exp(2*0.1256) = 1.2856	-35.2409
L_opt = exp(-3.41) = 0.0333	P_opt = exp(3.1944) = 24.3955	V_opt = exp(2*-0.1084) = 0.8051	11.9023
L_opt = exp(-0.3013) = 0.7399	P_opt = exp(-0.0013) = 0.9987	V_opt = exp(2*0.1731) = 1.4137	-32.0182

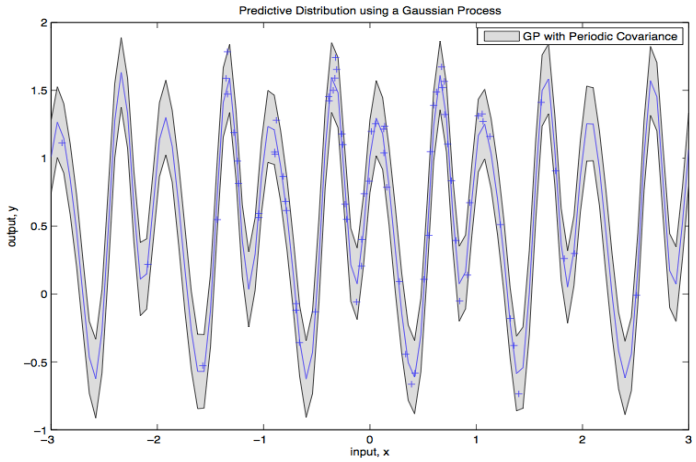


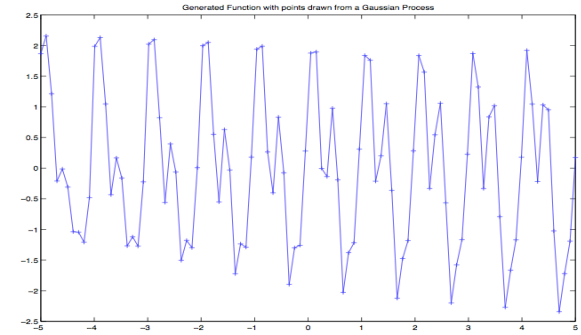
Figure 6 shows that the GP with periodic covariance fits the data perfectly, with high confidence, even in regions of fewer test points. As shown in table above, showing only a few optimized values, row 2 corresponds to lowest NLML with corresponding figure 6. This suggests that with appropriate initializations and optimized values, the data generating mechanism can be ideally periodic, which is also supported by the low negative marginal likelihood value.

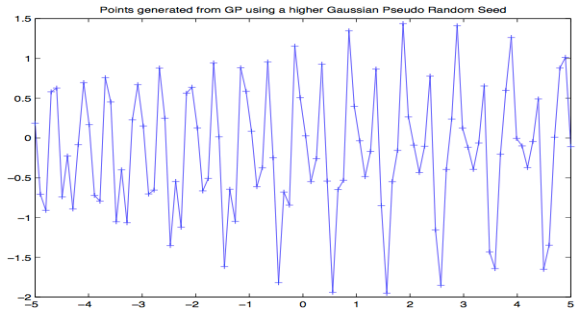
Compared to the fit in Question A, the model fits with higher confidence with very low uncertainty even in regions of less test points, suggesting that periodic kernels are a better choice of kernel compared to SE. The 95% error bars are lower, showing less uncertainty in predictions. With periodic covariance, local optimum of nlml can be achieved with value -35.24.

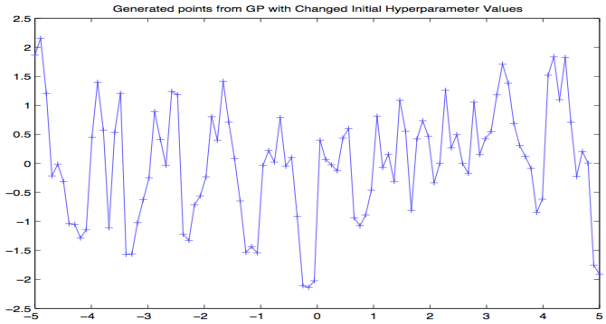
The trained GP with such kernels are a better fit with high confidence of the data generating process.

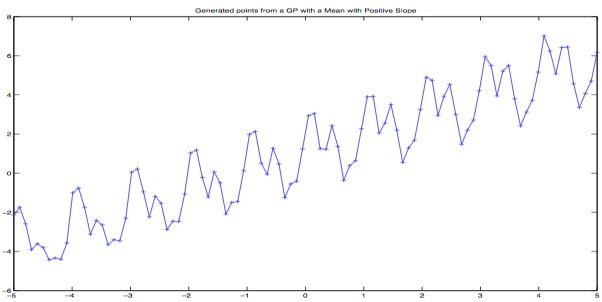
Question d

We need to add the small diagonal matrix because when doing the negative log marginal likelihood, or generate data points from a GP, we need to enforce constraints that the covariance matrix must be positive definite.
In figures below, we show example plots of points generated from a GP, with given hyperparameters. The plots below are for different sample draws from the GP, that we achieved by changing the seed of the random draw using gpml_randn function. In the last figure on column 2(right), we also show the effect of the sample function, when we add a linear mean to the GP model.







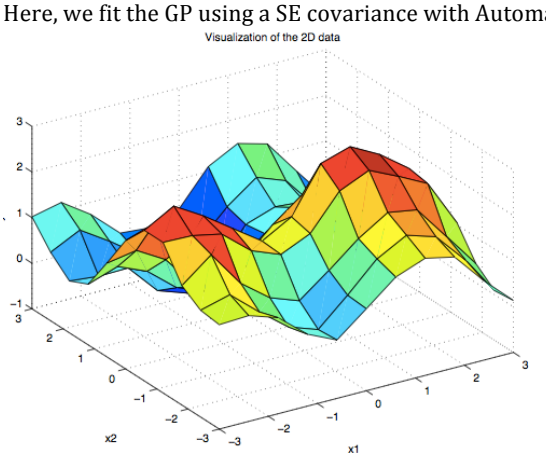


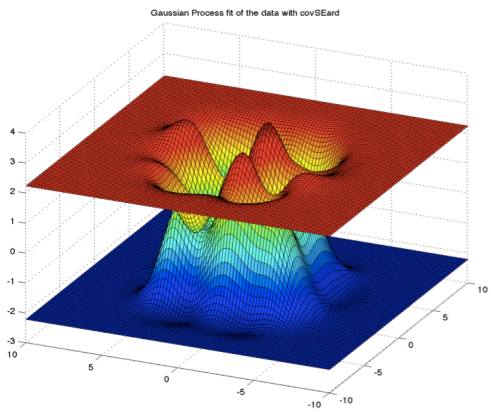
1st figure shows the periodicity in the generated data points due to the product of a periodic and squared exponential covariance function. A periodic behavior can easily be seen in the data. In 2nd figure in 2nd column, we have used a higher Gaussian pseudo random seed to observe the periodic behavior. Both these plots are for sampling from the GP.
In the 3rd figure (left), we have changed the initial hyperparamter values of the squared exponential, by using a very small length scale. Significant differences can now be observed in the generated data, where the SE now dominates the periodic behavior. Having a small length scale – indicating that the generated points are now more correlated in the input space. In 4th figure (right), we have added a positively sloped linear mean, observing the periodicity in generated data.

```
meanfunc = {@meanSum, {@meanLinear, @meanConst}};   covfunc = {@covProd, {@covPeriodic, @covSEiso}};
n=100;   K =  feval(covfunc{:}, hyp.cov, x);   K_I = K + 1e-6*eye(100);  %adding diagonal matrix      y = chol(K)'*gpml_randn(0.15, n, 1)
```

Question e

Here, we fit the GP using a SE covariance with Automatic Relevance determination. Figure below shows the 2-D visualizations.

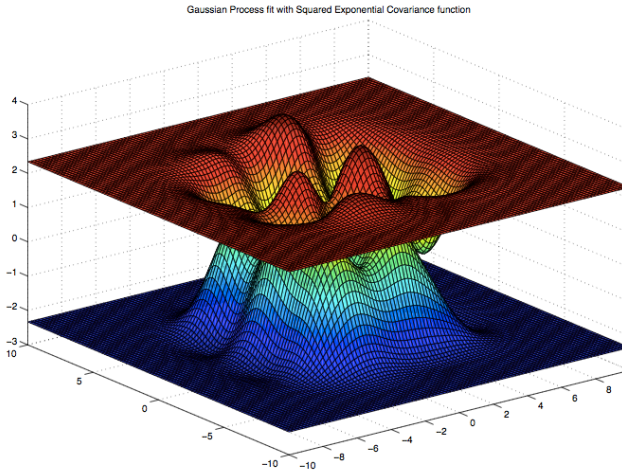




Using covSEard, with optimized length scale values of exp(0.4131) = 1.5115 and exp(0.2515)=1.2806, and variance of exp(2*0.1019)=1.2216, the NLML for the GP fit is (-) 19.2187. This result shows that even though the NLML supports the GP fit, having a low negative likelihood, but from the figure on the right, it can be seen that the GP fit is poor in regions away from the test points that we used. This is because we are using only one Squared Exponential along each dimension of the data set (ie, covSEard tries to fit the data along each dimension). The model fit is supported by the marginal likelihood. However, as we will see later, using symmetry breaking with additive covSEard functions gives a much better local optimum of the nlml.
We initialized the noise as 0.1, which has been optimized to a value of 0.1026 to fit the data. Therefore, there is not a lot of noise in the data generating process, using the covSEard covariance function.

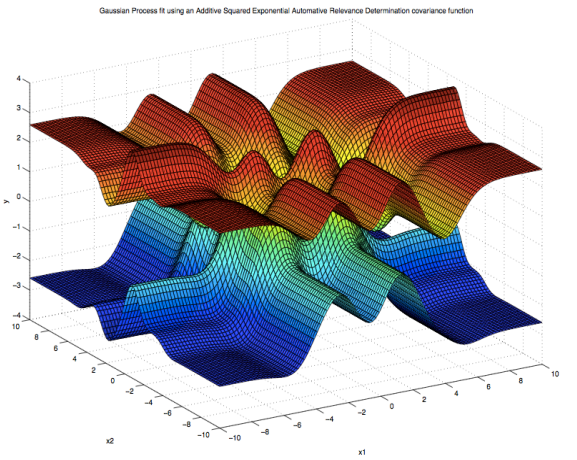
```
Xtest = linspace(-10, 10, 100);   Ytest = linspace(-10, 10, 100);   [L1,L2]=meshgrid(Xtest',Ytest');
 z =  [L1(:),L2(:)];       covfunc = @covSEard;    [ymu ys2 fmu fs2] = gp(hyp, @infExact, [], covfunc, likfunc, x, y, z);
```

Question f



With covSEard, the marginal likelihood was (-)**19.2187**; With using a squared exponential kernel, we get a **NLML=-18.0691** after optimizing the initial parameters, using a range of values.
The relative difference between negative log marginal likelihood is : **-19.2187 – (-18.07) = -1.1496.** The relative probability between the two models is therefore **exp(-1.1496) = 0.3168,** meaning that the model in Q.e is almost 3 times better than Q.f.
Since the NLML with SE kernel is lower (less negative), even though comparably similar, but it can be said that it is a poorer model compared to fitting the GP with a covSEard kernel. Additionally, using a range of test points from -10 to 10, similar to figure above, the covSEiso kernel does not generalize well to regions where there are no input data (as can be seen from the flattened portion in the graph). Code snippets for this are similar to above, except now we are using a covSEiso kernel.

Question g



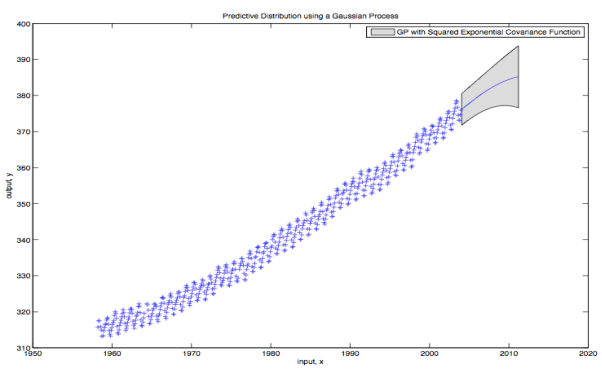
The additive covSEard kernels provide a much better predicted function to the data, having a NLML=-66.3365, showing that it is highly suggested by the marginal likelihood, and a better local optimum can be achieved using symmetry breaking. This has a much better fit to the data. Compared to Q.e and Q.f, this is a much better fit to the data, as can also be seen by the figure on the left. The predicted function now generalizes well in the test set region of (-10,10) having a confident predictive distribution. The wiggly regions in the plot shows that in regions where there are no input data points, it still can have a predictive function with certain uncertainty. Symmetry breaking is necessary for additive covSEards since now we are fitting two kernels along each dimension. The random initializations of hyperparameters is done such that the covariance function of the predictive distribution are centred along different points in the input dimension. An intuition for this is that – we are trying to fit two different distributions along each dimension (4 in total). If symmetry breaking not done, then assuming a Gaussian distribution, sum of two distributions would fit a single a Gaussian along each dimension making no difference to Q.e. Having symmetry breaking allows more representation of the predictive function along each dimension of input data.
Relative model probability with model from Q.e = exp(-66.3365 – (-19.2187)) = 3.4435e-21. With model from Q.f, relative prob is = 1.0918e-21
Code snippets: covfunc = {@covSum, {@covSEard, @covSEard}};
hyp.cov = 0.1*randn(6,1);
surf(L1,L2,reshape(ymu+2*sqrt(ys2),size(L1))); hold on;
surf(L1,L2,reshape(ymu-2*sqrt(ys2),size(L1)));

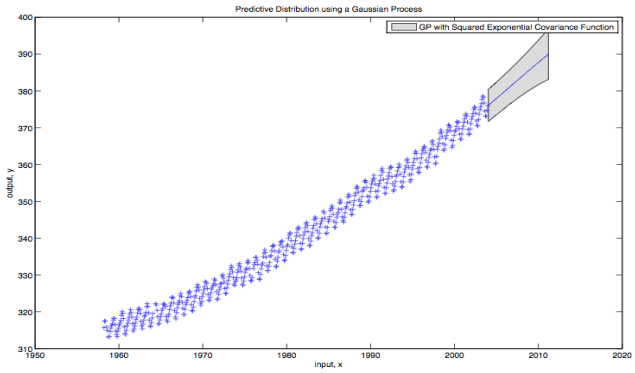
Question h

In this section, we add a linear mean to the GP, and fit the model using a squared exponential covariance function. Table below shows a segment of the different hyperparameter values tried to fit the data, along with the figures showing test set predictions. The high nlml values suggest that the GP data fit is not reasonable. Local optimum cannot be achieved and a GP with SE kernel is not modeling the periodicity in the data perfectly. Since nlml is a combination of data fit and complexity term, the GP fits below are too simple to model the data, resulting in the very high marginal likelihood values. The optimized value of the nlml are very high, reflecting a poor model fit to the data. Both plots below shows poor fit to the data, reflected by the high value of negative log marginal likelihood. The following code snippets were used:

L_init	L_opt	Var_init	Var_opt	Mean_init	Mean_opt	NLML
1.6487	12.73	2.72	15.20	[0.2 2]	[0.174 1.99]	1992
1.6487	0.5099	2.72	10.512	[5 10]	[0.1671 9.997]	963.38
12.72	18.71	15.18	22.93	[0.2 2]	[0.1761 1.9989]	1198

```
hyp.cov = [0.5; 0.5];   hyp.lik = log(0.1);    hyp.mean = [1; 1];   likfunc = @likGauss;
meanfunc = {@meanSum, {@meanLinear, @meanConst}};
```





All these fits to the data above are quite poor, as suggested by the high NLML values in table above. Even though from the plots seem to have good predictive distribution for the test points, such models are not favored by the NLML. As NLML is a characteristic of both data fit term and complexity, it might be that even though the data seems to fit well, such models are too complex for which the negative marginal log likelihood values are quite high.

Question i

At first, due to the periodic nature of the data, we add a SE and a periodic covariance function, with different initializations, and finding the local minimum of NLML. It can be inferred from previous observations, that there should a periodic kernel to fit the GP for this data. However, with covSEiso and covPeriodic, our local optimum NLML was 527.6, which is high, showing that the model is not obtaining a good predictive distribution. The plot below shows the result of adding the two kernels.

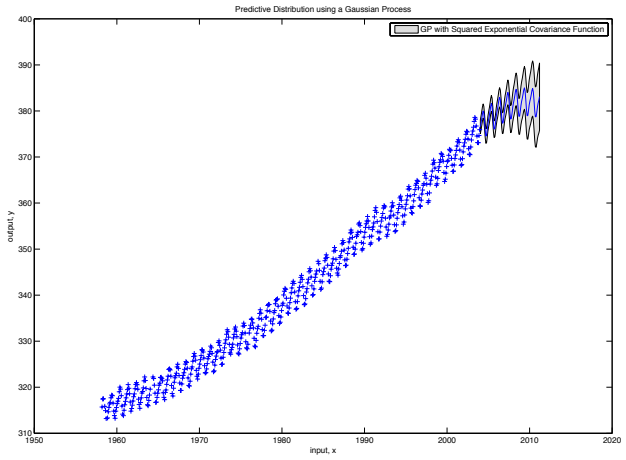


Figure 7: GP fit with Squared Exponential (SE) + Periodic (PER)

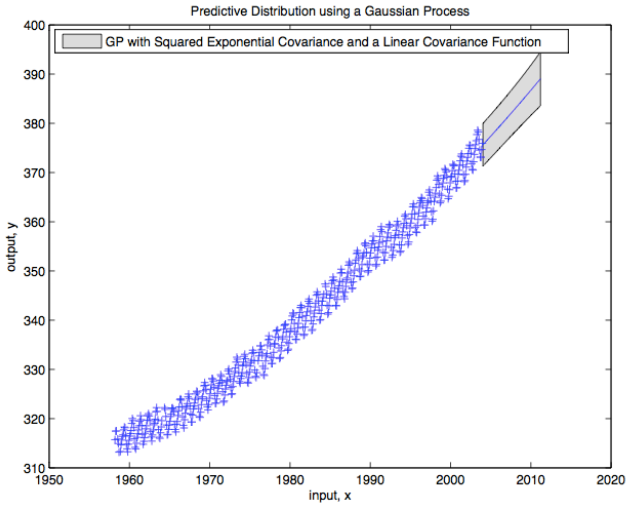


Figure 8: GP fit with SE + Linear (LIN)

Figure 8 shows a GP fit with a SE added to a Linear (LIN) kernel. With optimized hyperparameters, the local minimum is as high as 1201.6 showing an even poor fit to the data. The plot suggests that the model might be too simple for the data, such that too simple models are not supported by NLML.

Another possible covariance additive structure was to add two periodic kernels, since the data seemed to be periodic. However, such additive structures showed even poorer result (figure not included here due to space limitations), having a high NLML of 2354.5. The following code snippets were then used to combine the kernels together to generate results below:

```
k1 = {@covProd, {@covLIN, @covSEiso}};   k2 = {@covProd, {@covSEiso, @covPeriodic}};   k3 = {@covProd, {@covSEiso,@covRQiso}}; covfunc = {@covSum, {k1, k2, k3}}; hyp1.cov = [0.5; 0.5; 0.5; 0.5; 0.1; 0.1; 0.1; 0.5; 0.5; 0.1; 0.1; 0.1];
```

Using a combination of covariance functions below: (LINxSE) + (SExPER) + (SExRQ) – gives better local optimum of nlml

Finally, since it is clear that there is some linearity to the data, in addition to periodicity, and given that we needed a function values correlation too with the covariance structure, we decided to combine multiple kernels together. Here, we used a combination of 4 kernels (LIN, SE, PER and RQ (Rational Quadratic Covariance)) as follows:: (LIN x SE) + (SE x PER) + (SE x RQ) with a total of 12 hyperparameters that need to be tuned to obtain good predictive distributions. We also added a linear mean to the model. Figure 9 shows one of our plots with the above kernel combination having a **NLML of 296.71**. With further trial values of initial hyperparamteters, we got the further plot as shown in Figure 10, having a **NLML of 211.3366** (local minimum), and therefore such additive combinations kernels are much better GPs to model the data, than the ones considered before. The low nlml with optimized hyperparameters shows a good fit to the data, where model reflects both periodicity and linearity in the data.

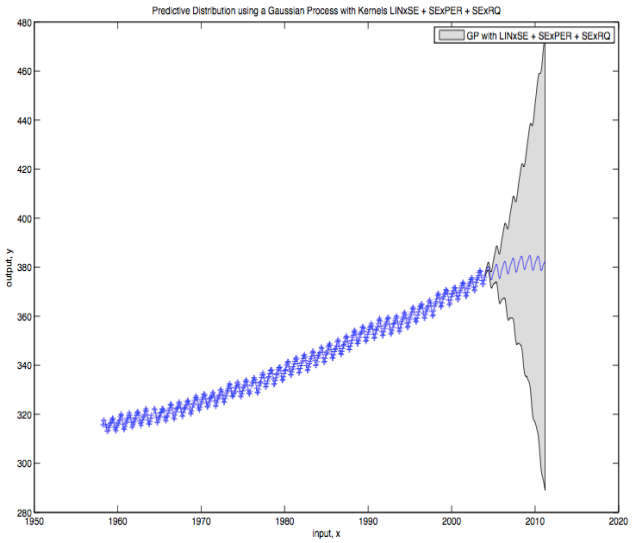


Figure 9: (LIN x SE) + (SE x PER) + (SE x RQ) with random initializations

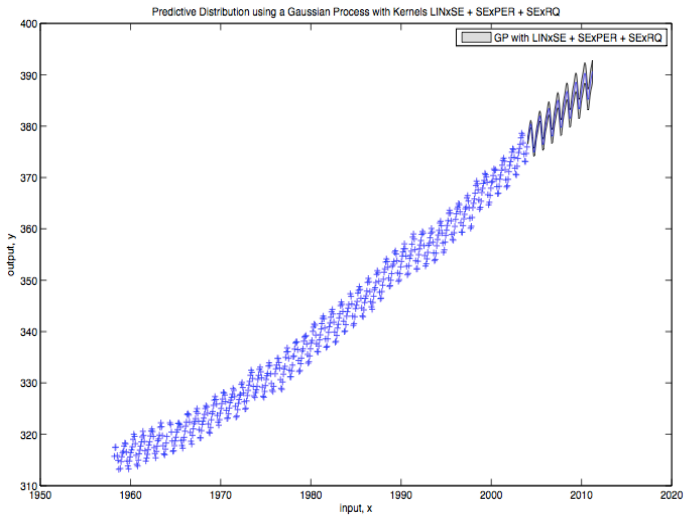


Figure 10: (LIN x SE) + (SE x PER) + (SE x RQ) with optimized hyperparameter values