# Report

The goal of this lab is to fit a sinusoidal function using gaussian radial basis functions.

We first build the training set. We take 75 random samples from 0.5 + 0.4 \* np.sin(3\*np.pi\*X) added with noise for x taken randomly from [0,1]. From the sample values of x, we need to model the gaussian RBF nodes.

To achieve this we first need to find clusters in the input x. We run k-means algorithm to find the k clusters and the k cluster centroids and label every training data to its closest cluster. We now have the center of the gaussian which is the cluster center. Next we need to find the width of the gaussian. This is achieved in 2 ways:

1. In every cluster, we find the variance of the points in the cluster. This becomes the variance of the guassian curve
2. Assign the same varianace to all the k gaussians as d2max/2k, where dmax is the max distance between the cluster centers.

The above two apporaches result in different results as can be seen later.

Once the cluster center and cluster variance is obained, we now can construct the k gaussian rbf nodes. The training data is fed to the gaussian nodes which result in transformed training data in a new space.

We then run a linear regression model on this transformed space. The weigh update is done according to the LMS update rule as follows:

w(n+1) = w(n) + eta \* [ d(n) - y(n)] x(n), where

d is the sample values and y is wT(gaussian output). Appropriate bias term is added to all the gaussian nodes.

We then calculate the predicted output for a range of x values, by feeding the x to the rbf nodes, then the output of the rbf network is scaled by the weights that were obtained and then summed to get the predicted output. We try to see if this model comes out as the sinusoidal wave that we initially defined.

Depending on the number of bases we have we observe underfitting and overfitting issues during prediction. When we chose variance of the clusters according to step 1, we see that k = 3 leads to best fitting and increasing the number of clusters overfits the data.

When we chose the variance from step2, we notice that k =16 best fits the sinusoidal curve. This is observed because when we take a common variance for all the clusters, we are basically increasing the coverage of gaussian over regions where there is less data support. Since we chose the variance as the distance between two farthest clusters, we are basically covering the entire length of the sinusoidal wave and hence more bases here will have the centers towards the end and hence the variance will cover all the data points.

We also notice that subtle differences in using the learning rate as 0.01 and 0.02. Eta =0.01 converges slowly as compares to eta = 0.02.

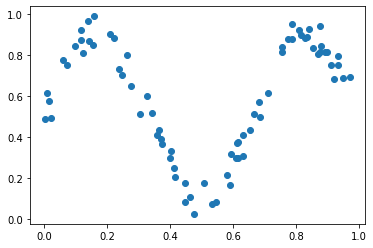
In places where there is missing data we find overfitting. THis can be seen in the multiple cluster variances part, where we see highs and lows.

It is also observed that the location of the cluster centroids determines the offset of the fitted sinusoid. For the best split when k=3, if we end up having clusters that are symetrical i.e divided equally in the 3 halves of the sinusoidal, the fitted function exactly matches the sinusoid. But in the unlucky scenario where we have assymetrical datapoints, we see that there is a slight offset in the fit sinusoid.

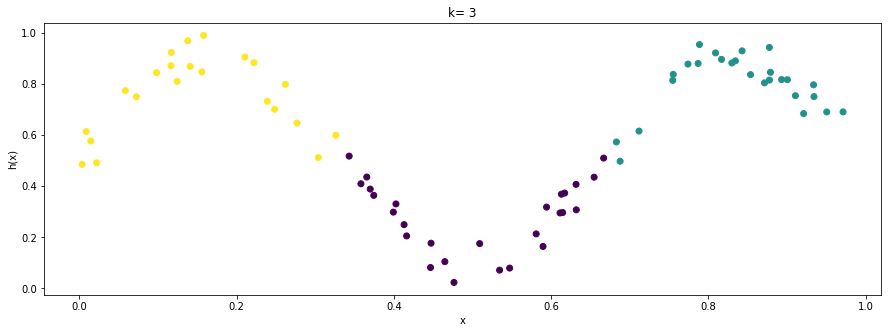
The advantage of one over the other comes into picture if we have a longer sinusoid. As shown in my results, when you want to fit a longer sinusoid, we need more number of bases. In case we are looking to fit a sinusoid of two cycles, we need approximately 16 bases with local variance to fit the sinusoid, but if we use the common variance method, we end up needing nearly 64 bases which makes things computationally expensive.

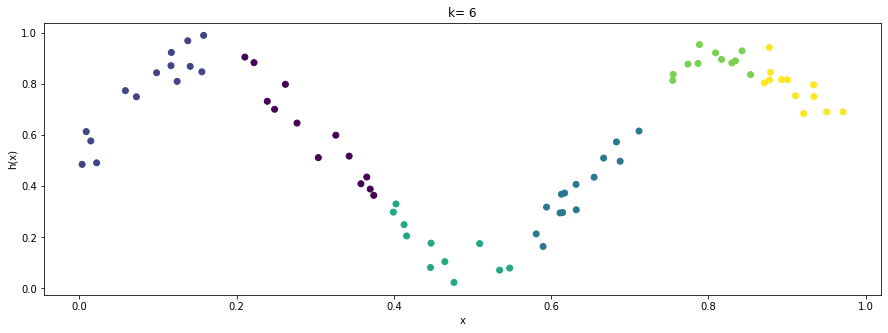
# Results

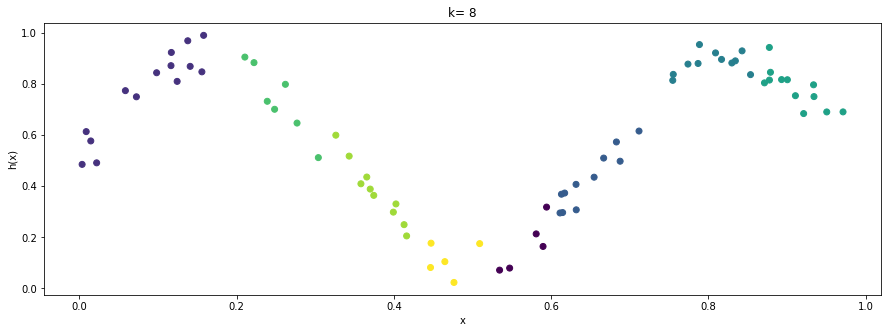
**Input data:**

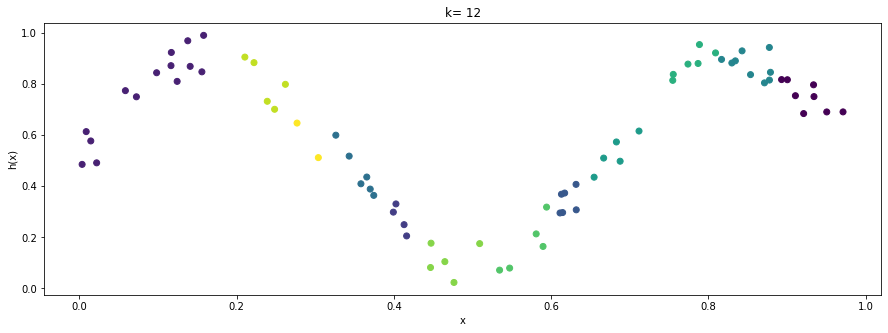


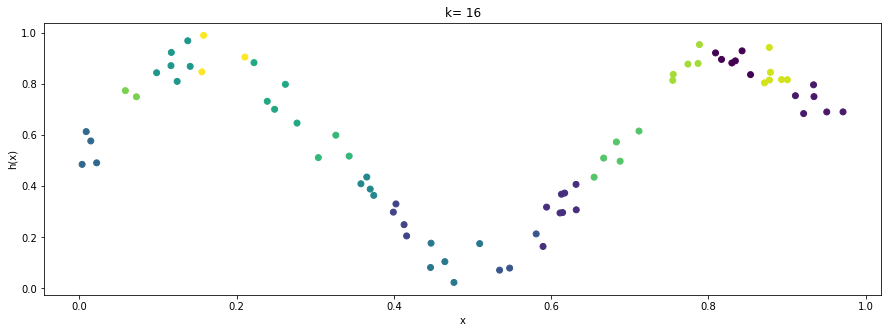
**Clustering according to k-means for k = 3,6,8,12,16**



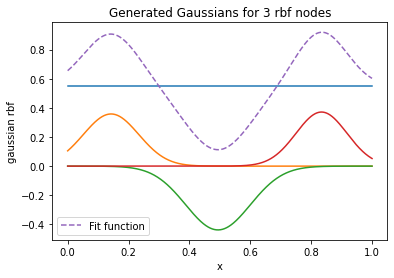


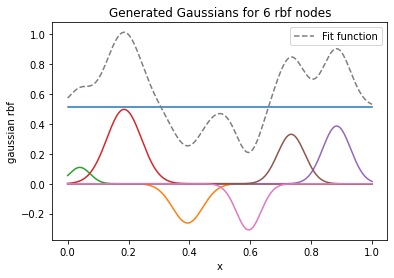


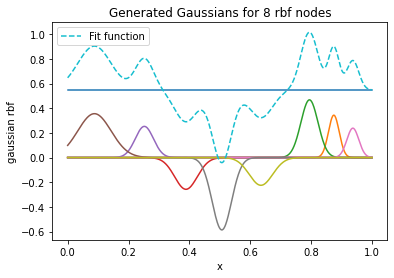


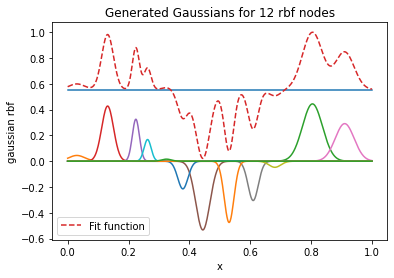


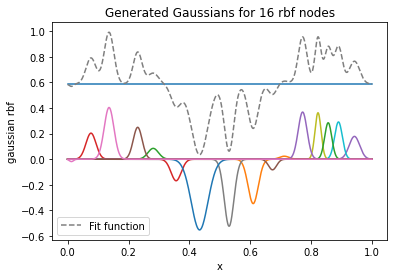
**Function fits using the guassian rbfs:**











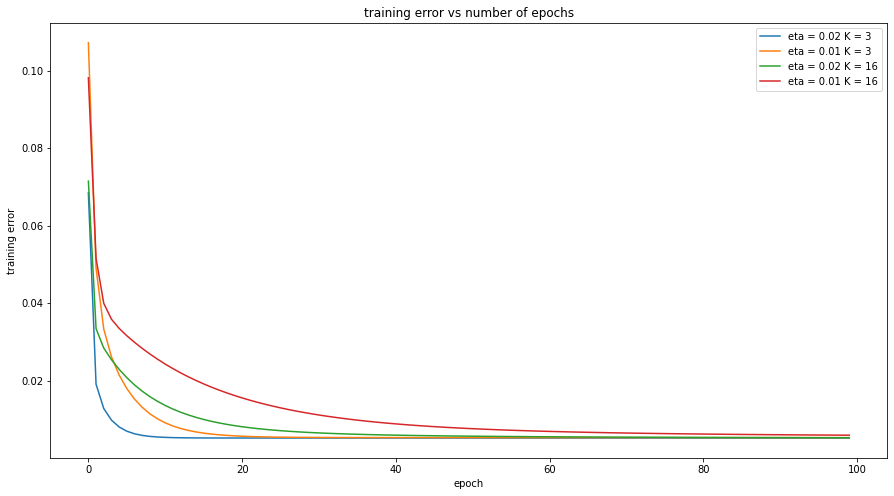
We see from the above graphs the different gaussian centers that have been generated from k-centroids. Also the gaussians have different widths, which can be attributed to the fact that we have chosen the variance of the gaussian as the variance of the points in the cluster

During training once the Gaussian RBFs are generated, we try to fit them with a linear regressor. Using the LMS update rule, we train the weights of the layer to correctly predict the hypothesis for the given training data.



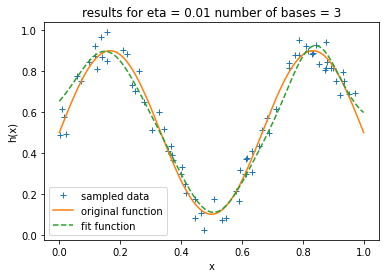
The above graph plots the training error vs epoch for the case of 3 rbfs with different variance. As can be seen there is not much difference between the two, although eta =0.02 converges faster than eta=0.01

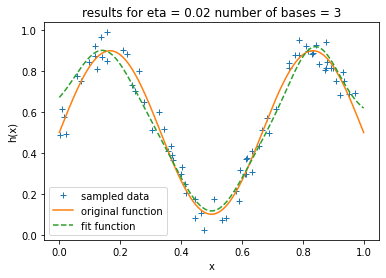
But when we increase the number of rbfs, we see that the network takes longer time to converge. This can be seen for the below graph for 16 rbf nodes.



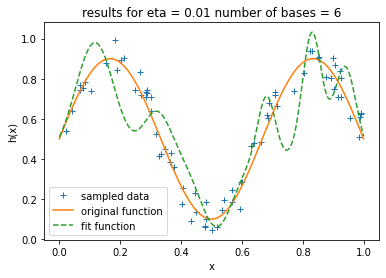
This is because increasing the number of bases leads to wrong classification many times.

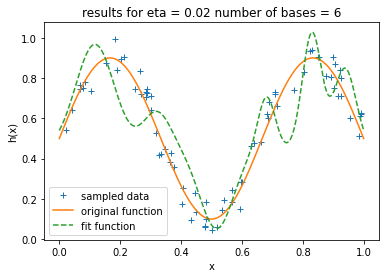
**K = 3**



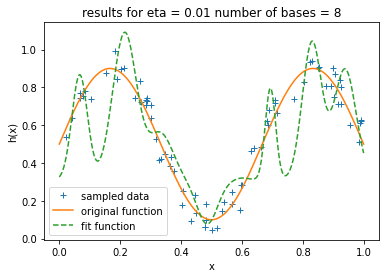


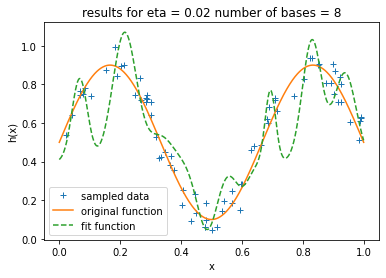
**k=6**



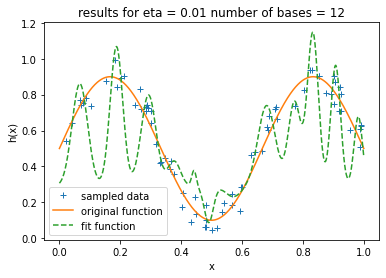


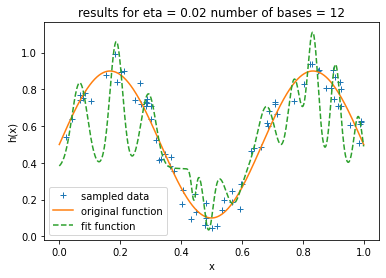
**k =8**



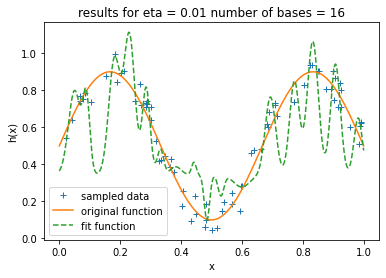


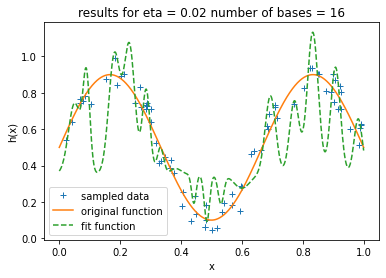
**k=12**





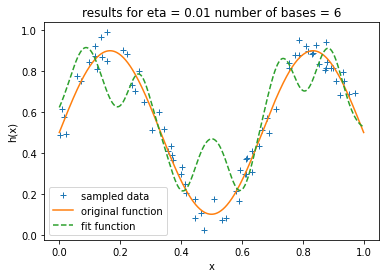
**k=16**

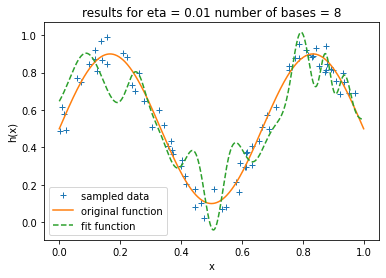


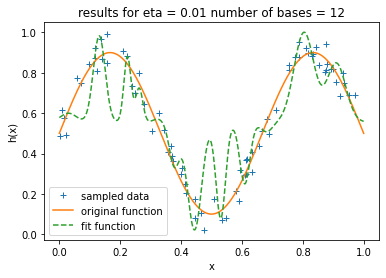


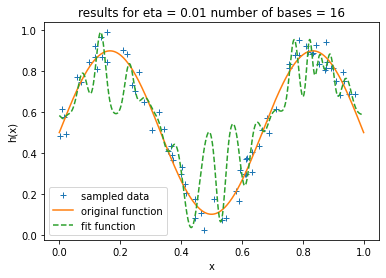
As can be seen above there isn’t much difference for the two etas in the fitted function. Also number of bases =3 has correclty fit the function. Although there is some error in cases where there is lack of data support and also in the beginning and the end of the graph.

**Overfitting issues**



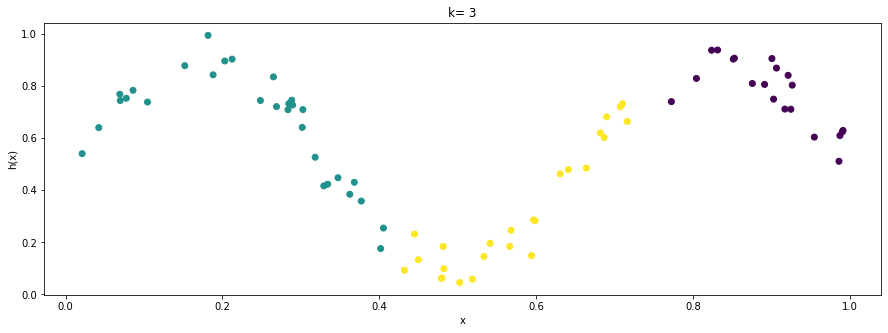


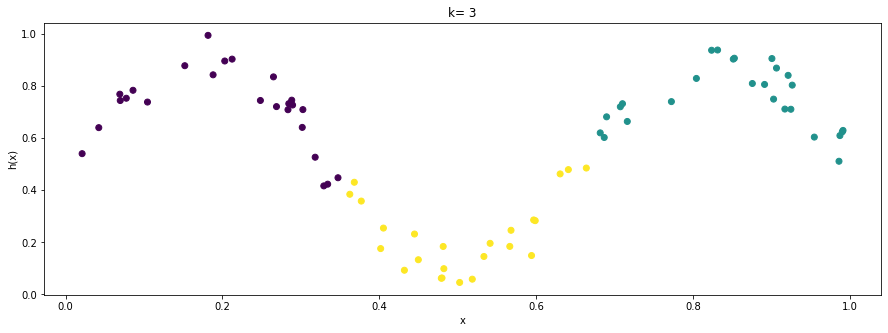




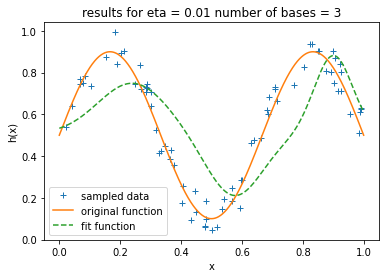
We see peaks of data in all the 4 above plots. This is a variance/overfitting problem. As the number of bases increases we have lots of gaussians for a point, and if there is no such point, it becomes really difficult to predict the output.

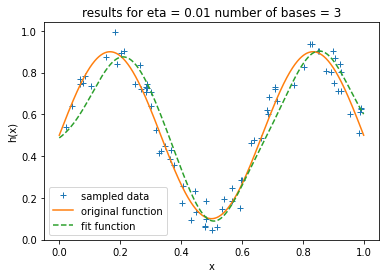
**Effect of having different cluster centers**





From the above two graphs we see that in case of the second graph, we got lucky with the split of points. So we have a symetrical split in data points here. So in this case, we get a very good fit as compared to case1.

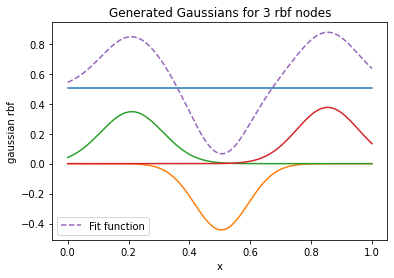




The above plots were the results of different cluster centers for the same number of basis. As can be seen the second plot is well fit compared to the first because we got a lucky symetric split of points. And hence the gaussians are spread across symettrically and when we add them up, we end up with the exact split.

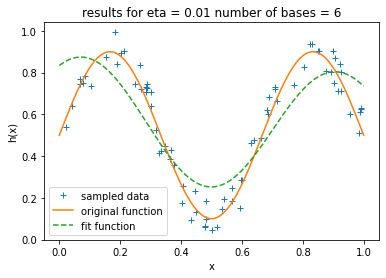
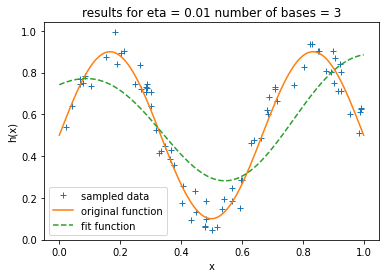
**Gaussians with same variance for all clusters**

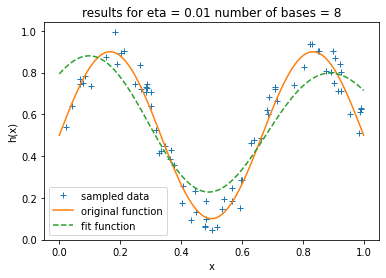
Having common variance across all the RBF nodes seems to have fixed the issue of having extra peaks in the graph.

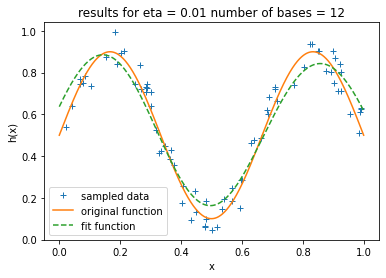


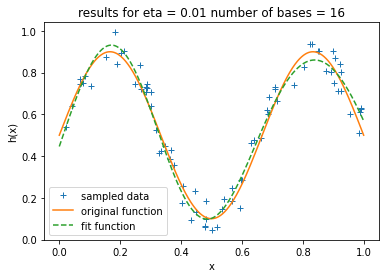
The above graph shows the gaussians generated for 3 bases. It can be seen that they all have the same width.

**Function Fitting**



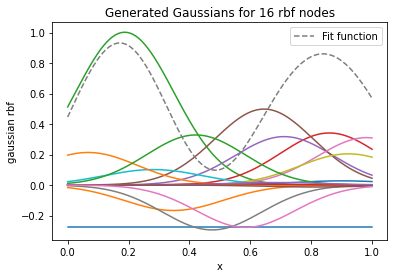


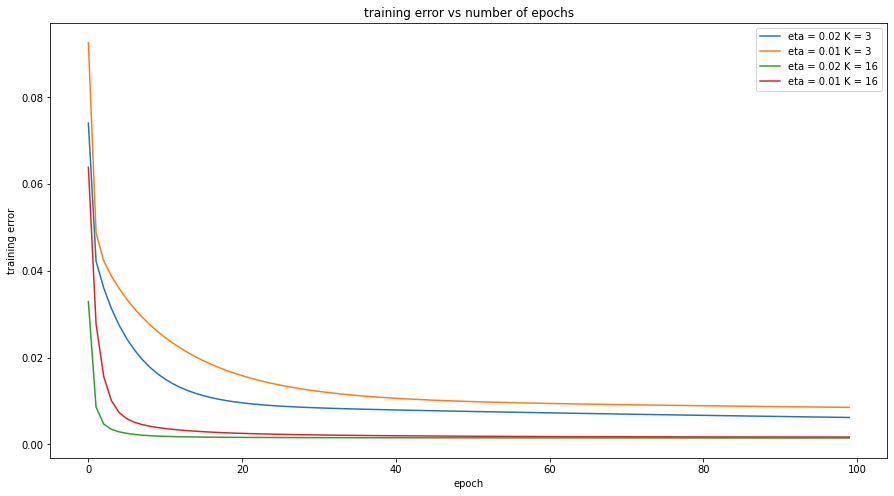




As can be seen from the above graphs as k increases we get better approximations and for k =16, we have the best fit for the sinusoidal function.

In this scenario, the linear regressor is able to learn all the weights appropriatley for the 16 bases as the width is spread over all the areas of the curve as shown below:

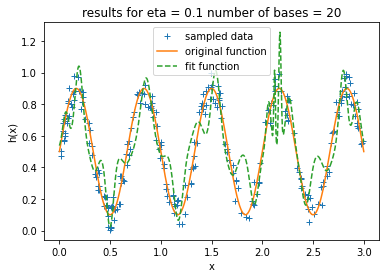




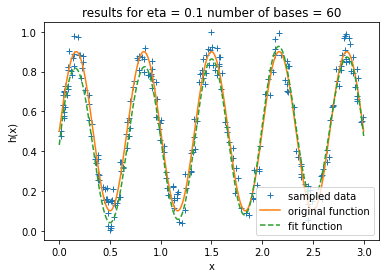
We can also see now that k=16 converges faster than k=3 and eta 0.02 converges faster than eta 0.01

**3 cycle sinusoid**

Local variance considered



Common Variance across all the bases.



We need more bases in case we use the common variance to fit the data. So this might make things computationally expensive when we can get the best fit with less bases in case 1.