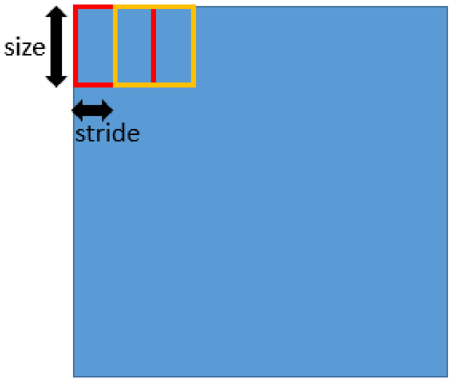
# HW-1: Linear Regression

# Final Report

- Introduction

In this homework, I divide whole map into small regions. Here comes out a parameter called region size. In addition to the size of the local region, I should decide whether these regions are overlapping with others. And, it's decided by the term stride.



At the beginning of my process of training model, I should set size and stride. Then the small square filter will scan whole map. In each local region, I design to put a 2 Dimension Gaussian distribution there. Each Gaussian function is a basis function. However, we can see that 2-Dimension Gaussian function is an exponential form with some important parameters such as $\mu_x$,$\mu_y$, $\sigma_X$and$\sigma_y$. Then, I am going to show how I make those parameters.

For mean value in x and y coordinates of each local region, I use the concept of Weighted Average, which makes sense. For example, we should put the center of 2 Dimension Gaussian distribution near the peak of height map and it can be approximately done by weighted average. In our model, the weightings are their height which we can get from training data set. And I assume $\mu_x$ and $\mu_y$ are independent. Thus, I can do the simple Weighted Average in each coordinates.

On the other hand, it is better to train our model with different variance for each local region. Nevertheless, to make it simple calculating and after trials, I design the sigma as same as size of local regions to make each Gaussian distribution to greatly overlap with others, which adds more flexibility to our training model. If we make each of them overlap with small percentage, our model is weak to present the data point at the boundary between these two Gaussian distribution. After setting for these for parameters, we can do the same process for next local region.

- Approaches

  In each approach, the setting and the whole process are the same as above. The only obvious difference is how the parameters of linear combinations are obtained. The following I am going to briefly introduce each solving process.

  - ML approach
    - Introduction to my ML predictor

      Assume the data points are independent, and we can obtain the likelihood function with parameter w and $\beta$ as the form of product of normal distributions. The next step is to maximize the likelihood function. Equivalently, we can minimize a sum-of-squares error function between target value and $w^T\varphi(x_n), n = 1, \dots, N$..

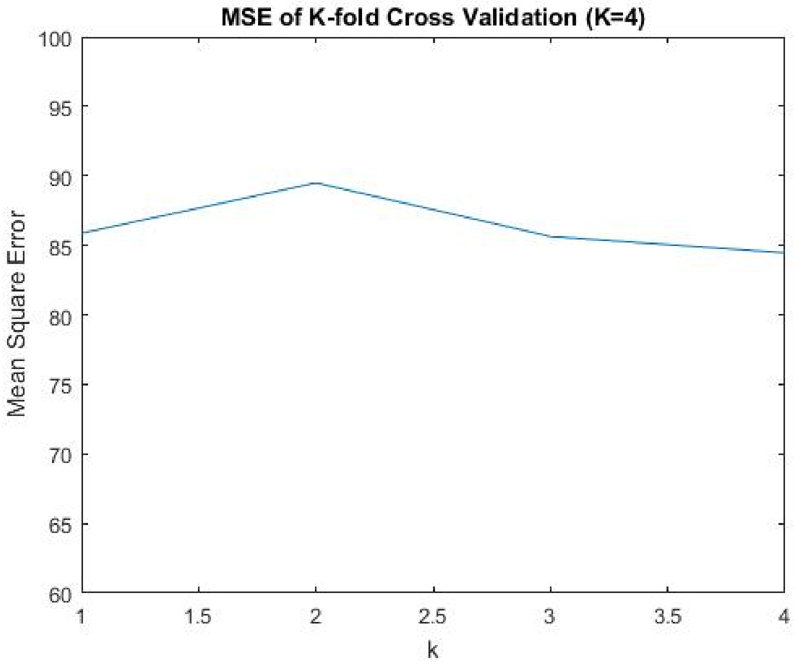
      $$\mathrm{P}(\mathrm{t}|\mathrm{X},\mathrm{w},\beta)=\prod_{n=1}^{N}\mathcal{N}(t_n|w^T\Phi(x_n),\beta^{-1})$$

      After solving for w, we obtain $W_{ML}$ and take it to the equation y(x,w)= $w^T\varphi(\mathrm{x})$ to acquire the estimation.

    - K-fold Cross-validation:

      The reason why we have to do K-fold Cross validation is to evaluate estimator performance. If we just learn the parameters through same training data and test the estimator on the same test data, the overfitting will likely to happen. Thus, a solution to this problem comes out. We can split the training data set into k smaller sets and use one small set for each validation.

      In my homework, I use 4-fold Cross Validation and the following graph shows each MSE. We can see that the MSEs is almost the same with little error, which means that my estimator is good and does not favor any small set of training data.



- ◆ Result (Model Performance)

  Data Set Information:

  Given 40000 training data, I divide it into two set.

  Training data: 30000

  Validation data: 10000

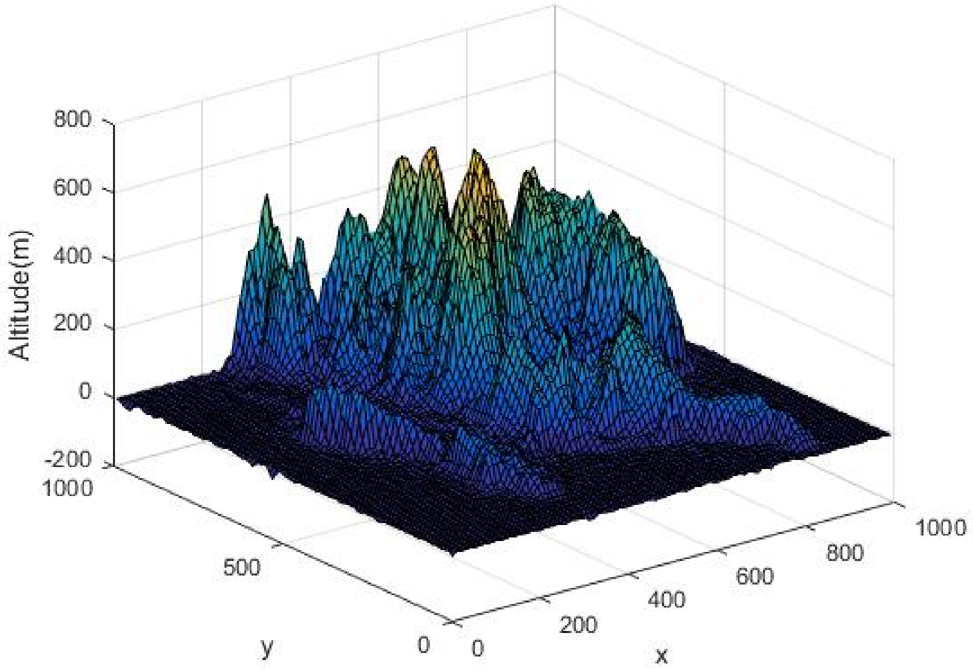
  Local Information:

  Size: 25

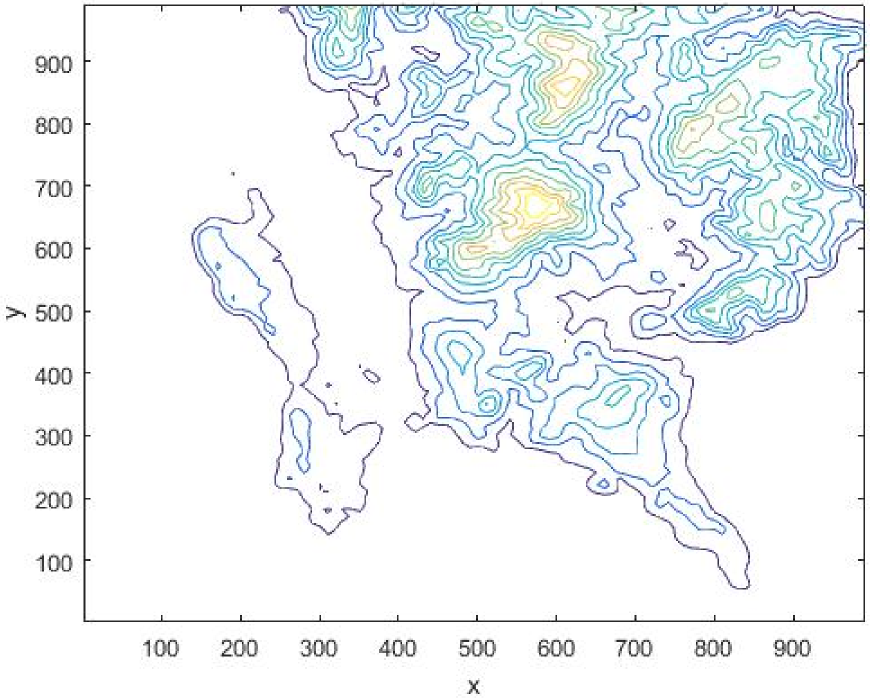
  Stride: 13

  ⇨ MSE: around 85~90 in 4-Fold Cross-Validation

- ◆ Visualization of Height map
  - ● 3-Dimension



- 2-Dimension



- MAP approach
  - Introduction to my MAP predictor

    Basically, MAP is the same as ML. However, the only difference is that MAP introduces a prior distribution. At the same meaning, it add a regularization term to its error function to control overfitting. In its regularization term, it relates to lambda. When lambda is too high, the resulted model has poor performance; On the contrary, when lambda is too low, the model is likely to exist overfitting.

- Result (Model Performance)

Data Set Information:

Given 40000 training data, I divide it into two set.

Training data: 30000

Validation data: 10000

Local Information:

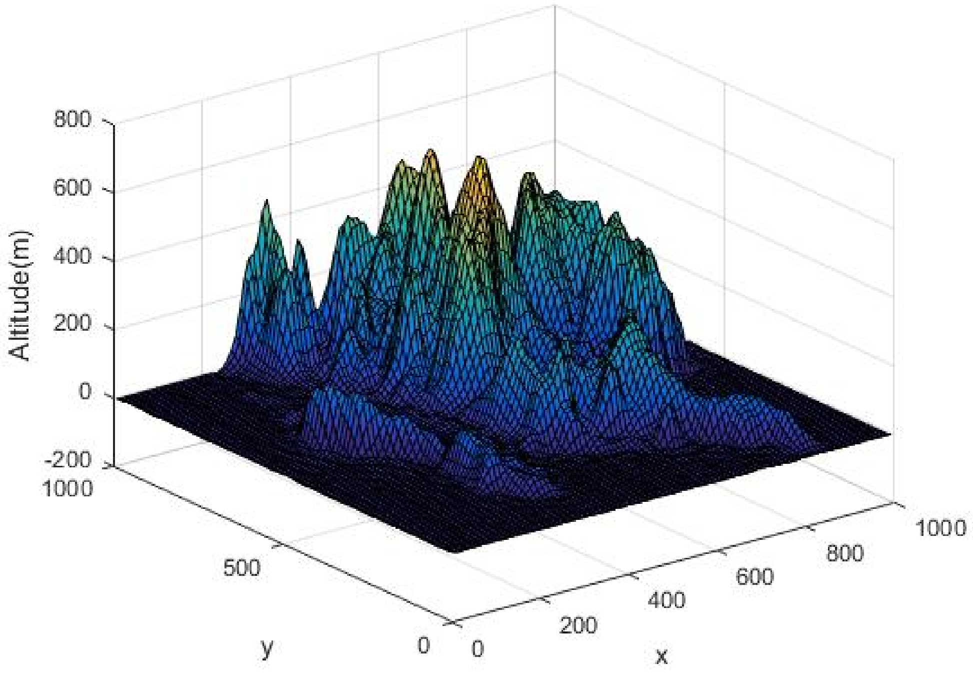
Size: 25

Stride: 13

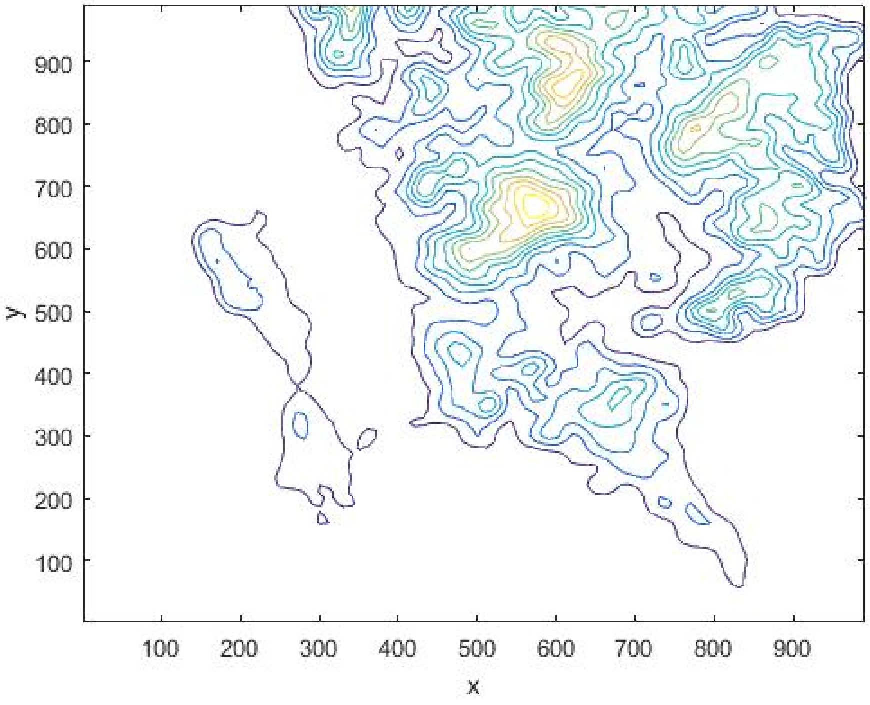
| Lambda | 0 | 1e-5 | 1e-3 | 1 | 3 |
|---|---|---|---|---|---|
| MSE | 120.50 | 142.74 | 177.71 | 297.26 | 335.09 |

From above, as lambda value decrease, it toward well-fitting, which introduce more flexibility into the training process. Besides, the reason why the MSE value at zero lambda is higher than the result of ML approach is due to different matrix calculations but still with same meaning. However, those errors are tolerable. .

- Visualization of Height map
  - 3-Dimension



- 2-Dimension



- Bayesian approach
  - Introduction to my MAP predictor

    The above two approaches leave the issue of deciding the proper model complexity. Bayesian approach, which will lead to automatic determining model complexity, solves this problem

According to the proof in the textbook in 2.3.1、2.3.2, we can conclude that a Gaussian p(x) in which we divide into two subsector $(x_a, x_b)$. We noted that the mean of the conditional distribution p($(x_a|x_b)$ was a linear function of $x_b$.

Then, we make an assumption

$$\mathrm{p}(\mathbf{w}) = \mathcal{N}(\mathbf{m}_0|\mathbf{S}_0)$$

Besides, we have already know

$$\mathrm{p}(\mathbf{t}|\mathbf{w}) = \prod \mathcal{N}(\mathrm{t_n}|\mathrm{y}(\mathbf{x}_\mathrm{n}, \mathbf{w}), \beta^{-1}) = \prod \mathcal{N}(\mathrm{t_n}|\mathbf{w}^\mathrm{T}\phi(\mathbf{x}_\mathrm{n}), \beta^{-1})$$

Therefore, we can use the proof deduced in the textbook that the mean of p(t|w) is linear combination of w. As our model is Gaussian function, I make the corresponding conjugate prior as also Gaussian function. Note that it is easily concluded that the result product of prior probability distribution and likelihood function, posterior distribution p(w|t)= $\mathcal{N}(m_N, S_N)$, is also Gaussian function. It's peak occurring at it mean vector $m_N = W_{MAP}$.

Moreover, to make it simple, the prior distribution has zero mean and one parameter $\alpha$. And according to this setting, our posterior distribution has $m_N = \beta S_N \Phi^T t$ and $S_N^{-1} = \alpha I + \beta \Phi^T \Phi$. Then take the logarithm, we would have the following equation.

ln p(w|t) = $-\frac{\beta}{2}\sum_{n=1}^{N}\{t_n - w^T\varphi(x)\}^2 - \frac{\alpha}{2}w^T w + const.$

To get the maximum of posterior distribution, we can instead do minimization of error function with regularization term with lambda using $\alpha/\beta$. Coincidentally, this minimization is the same as MAP approach.

From the perspective of predictive distribution, we would have the following equation. Also, based on the proof above, we can have further equation which is equivalent to MAP.

$$\mathrm{p}(\mathrm{t}|\mathbf{t}, \alpha, \beta) = \int \mathrm{p}(\mathrm{t}|\mathbf{w}, \beta)\mathrm{p}(\mathbf{w}|\mathbf{t}, \alpha, \beta)\mathrm{d}\mathbf{w}$$

$$= \int \mathcal{N}(\mathbf{w}^\mathrm{T}\phi(\mathbf{x}), \beta^{-1})\mathcal{N}(\mathbf{w}|\mathbf{m}_\mathrm{N}, \mathbf{S}_\mathrm{N})\mathrm{d}\mathbf{w}$$

$$= \mathcal{N}(\mathrm{m}_\mathrm{N}^\mathrm{T}\phi(\mathbf{x}), \frac{1}{\beta} + \phi(\mathbf{x})^T \mathrm{S_N}\phi(\mathbf{x}))$$

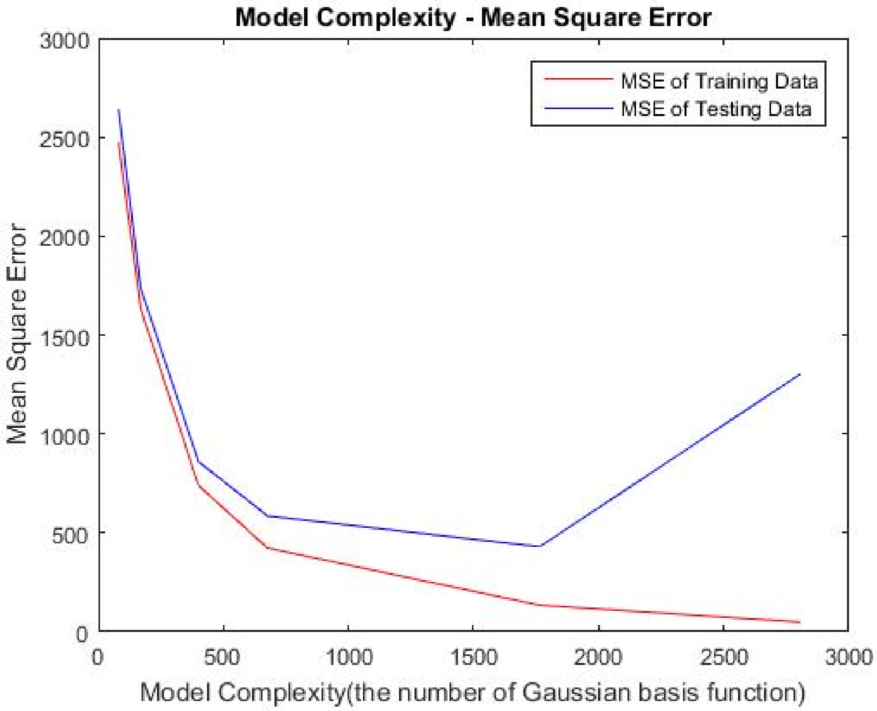
In conclusion, if it's Gaussian prior, we can say that Bayesian approach equals to MAP approach.

- Result (Model Performance) & Visualization of Height map

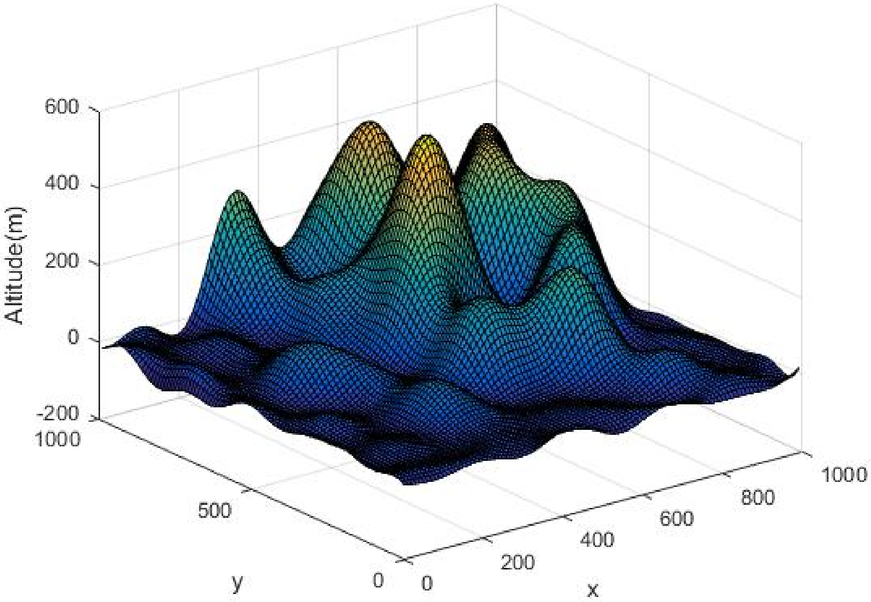
  Same as the result in the MAP part.

- Overfitting & Underfitting

  Overfitting literally means over training. In other words, it means that the hyperplane is biased. It is excessively close to training data. The condition in which overfitting is likely to happen when the model complexity is too high. In our case, the number of Gaussian basis functions are realized as model complexity. Since it requires a powerful computer to overfit 30000 training data, I would like to rearrange my training data set. Also since the training data are randomly given, I pick the first 5000 training data to be my new training data set. Thus, I can easily achieve overfiiting training data with ML approach. The following graph shows that each MSE would decrease as model complexity increase. But after about 1700, the MSE of training data still decrease while the MSE of testing data is increased quickly.

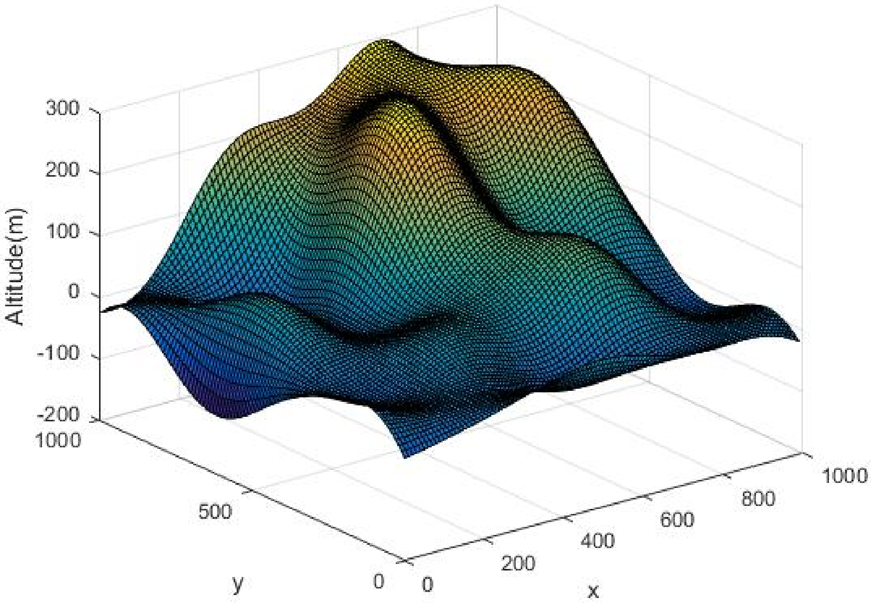


On the contrary, underfitting refers to a model that cannot well present the distribution of training data. And, we can easily detect underfitting when the MSE of training data is big compared to the well-trained model. As the following two 3 Dimensional graphs show, the first one has only 169 Gaussian basis functions and the linear combination of them is bad at predicting detail part of the height map. Also, its MSE of training data equals to 1762, meaning that around 41.9 meters error at each data point.



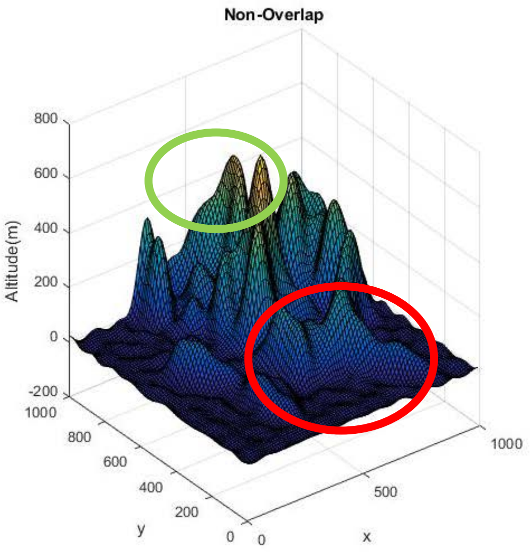
.

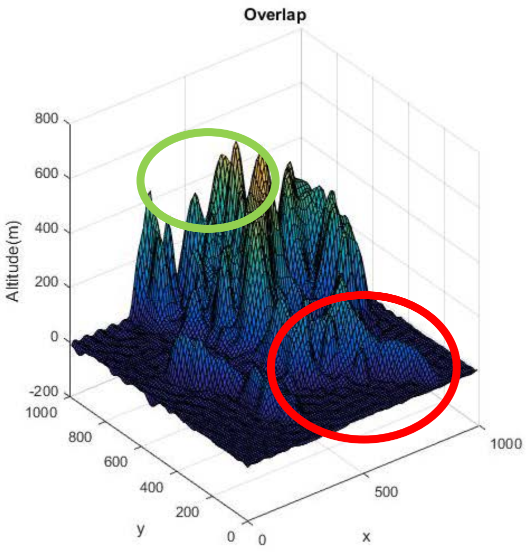
To be more underfitting, I reduce the number of Gaussian basis function as 64 which is really small number compared to the size of training data set. Besides, the MSE of training data equals to 3279, meaning that 57.2 meters error at each data point, which is really worse.



- Discussion
  - Drawback of non-overlapping local region

If we set the size of the local region as 50 and stride as also 50, in other words, non-overlapping, it will has less flexibility to describe the data point near the boundary of the local region, which is demonstrated below. Then, I make the stride as the only difference in my experiment to see the performance of established model. One without overlap has greater MSE than the one with overlap. Thus, I would like to overlap my designed local region to achieve better performance.





■ Variance setting

In this part, I am going to discuss the variance for each Gaussian basis function. Under the setting with stride 25 and size 50, I run the file to get each MSE for different sigmas. The following table is the result MSE for corresponding sigma. In my opinion, I guess the relatively higher MSE with sigma small than size results from the confinement of Gaussian distribution. On the other hand, with relatively higher sigma, we can think one extreme case that the distribution is extended to uniform distribution, which seems inflexible to simulate the distribution of training data. And I am sorry for that I have not taken the ‘Detection and Estimation’ course yet and also not found any useful document to prove the idea above.

| sigma | 25 | 50 | 75 | 100 |
|---|---|---|---|---|
| MSE | 276.64 | 235.60 | 384.97 | 606.35 |

● Reference

[1] Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer, 2007