Machine Learning 2017

HW1: Linear Regression

A052153 洪章瑋

Linear regression model

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

In this homework, your model should be implemented by the feature vector, which is defined:

$$\boldsymbol{\phi}(\mathrm{x}) = [\phi_1(x), \phi_2(x), \ldots, \phi_N(x), \phi_{bias}(x)\,]$$

y(x, w): prediction of linear model, $y \in R$

$\phi(x)$: feature vector of x, $\phi(x) \in R^m$

w: weight of each component, $W \in R^m$

Basis function

Since Gaussian basis is suit for modeling height map of natural terrain, I choose Gaussian basis function as basis function.

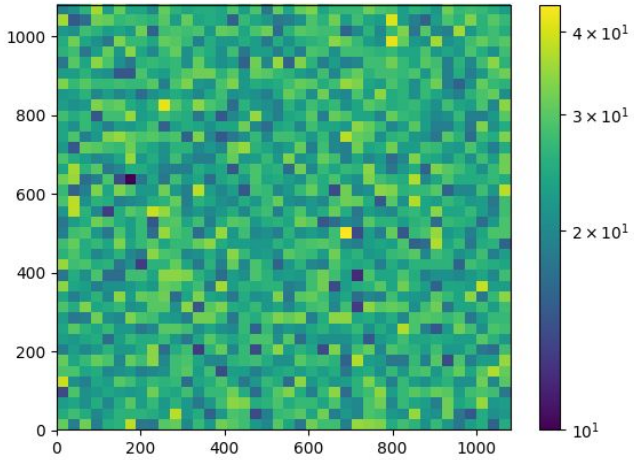
2D Gaussian basis function is defined as the following formula.

$$\phi_n(x) = \exp\left(-\frac{(x_1 - \mu_i)^2}{2s_{1n}^2} - \frac{(x_2 - \mu_j)^2}{2s_{2n}^2}\right), for\ 1 \le i \le O_1, 1 \le j \le O_2$$

The reason that why I don't choose Polynomial, Sigmoid as basis function is that both of their distribution are not suit for modeling terrain.

By using nonlinear basis functions, we allow the function y(x, w) to be a non-linear function of the input vector x.

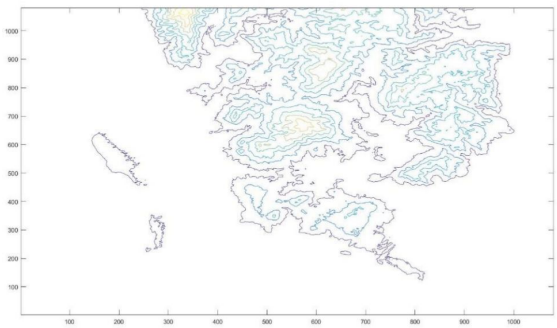
Data preprocessing



(Fig.1 distribution of training data)

According to Fig.1, it is obvious that the training data uniformly distribute in the input space. It is unnecessary to worry about unbalanced sampling in training data. (i.e. I can directly shuffle training data and split into training set, testing set).

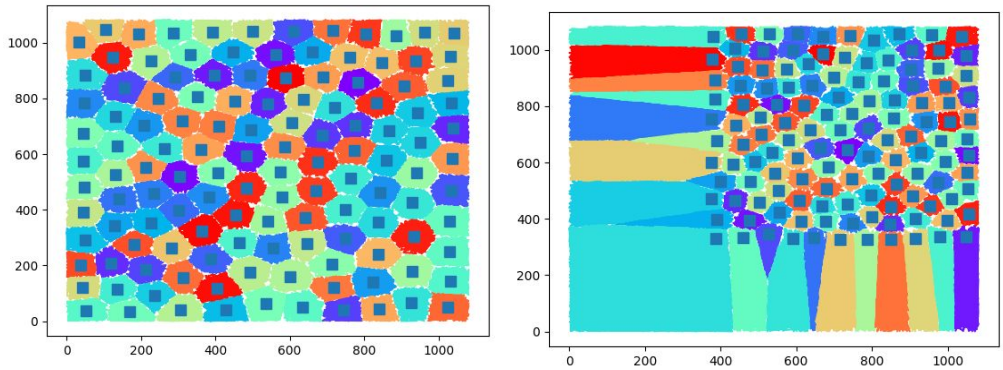
Since using Gaussian basis requires pre-defined mean, sigma, I experiment two approaches to find suitable mean and sigma. There are two versions of both approaches. In the first version, I calculate mean and sigma on all training data. In the second version, I filter out some unimportant data according to my observation of data (Fig.2) because the height is almost uniform around some region in the map.



(Fig.2 ground truth of height map)

K-means

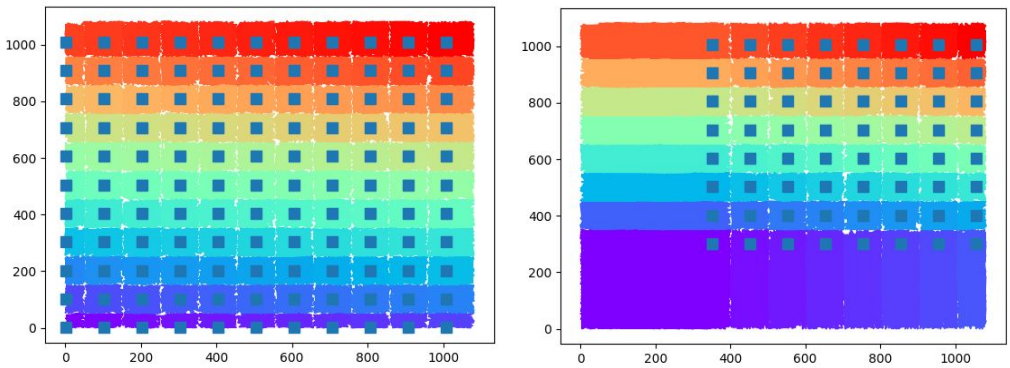
The intuition of using K-means is that I want to cluster the training data into K cluster. And take those centroids of cluster as means and cluster size as sigma. The distribution of centroids are illustrated as the following two figures.



(Fig.3 (Left) no filtered (Right) filtered)

Quantize

Since heavy computation overhead of K-Means when dimension increasing, to experiment with higher dimensional feature vector, I take an alternative approach - Quantization.
In this approach, the blue points in the following figures are mean values in basis function and the distance between two blue points is sigma.



(Fig.4 (Left) no filtered (Right) filtered)

Maximum likelihood

In Maximum Likelihood approach, objective function of optimization process is:

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2.$$

(Eq.1)

I apply stochastic gradient descent (SGD) to minimize this function instead of solve w by Normal equation due to poor performance (not computation efficiency). The experiment result will be mentioned in later section.

However, to compute on GPU, I modify original SGD to batch stochastic gradient descent. Instead of using only one sample from training data, my batch version train the model with mini-batch of training data which is more data-efficient.

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n$$

(Eq.2)

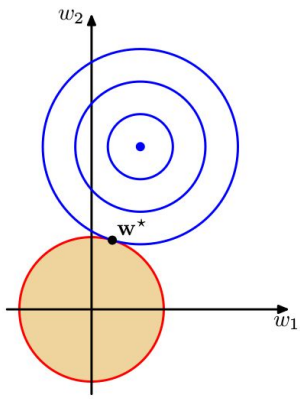
Maximum a posterior

Maximum a posteriori (MAP) approach is almost same as Maximum Likelihood (ML) approach. The only difference is that in MAP, I take prior into account. The prior in the objective function can be taken as Regularization term. The objective function is:

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

(Eq.3)

Same as ML approach, I use SGD algorithm to minimize this function. With Regularization, overfitting can be avoided.



(Fig.5 Ridge regularzation)

Bayesian approach

In Bayesian approach, instead of estimate only Likelihood, estimate product of likelihood and prior simultaneously.

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

Eq.4

And Eq.4 can be integrate analytically and rewritten as:

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

Eq.5

And those variable in Eq.5 can solved by Eq.6:

$$\begin{aligned} \mathbf{m}_N &= \mathbf{S}_N \left(\mathbf{S}_0^{-1} \mathbf{m}_0 + \beta \mathbf{\Phi}^{\mathrm{T}} \mathbf{t} \right) \\ \mathbf{S}_N^{-1} &= \mathbf{S}_0^{-1} + \beta \mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi}. \end{aligned}$$

Eq.6

Not like ML and MAP, in Bayesian approach, there is no necessary to use SGD to solve *w*.

Experiment

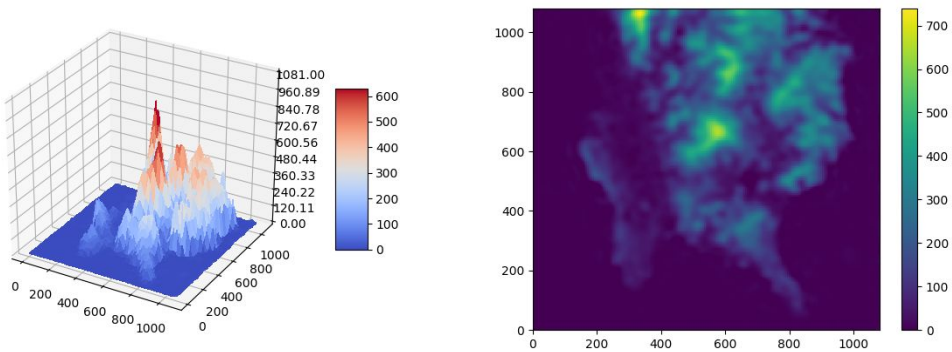
Setup

Training data is split into training set and testing set. (80% are training set, 20% are testing set). The following hyper parameters are selected according to K-fold cross validation result.

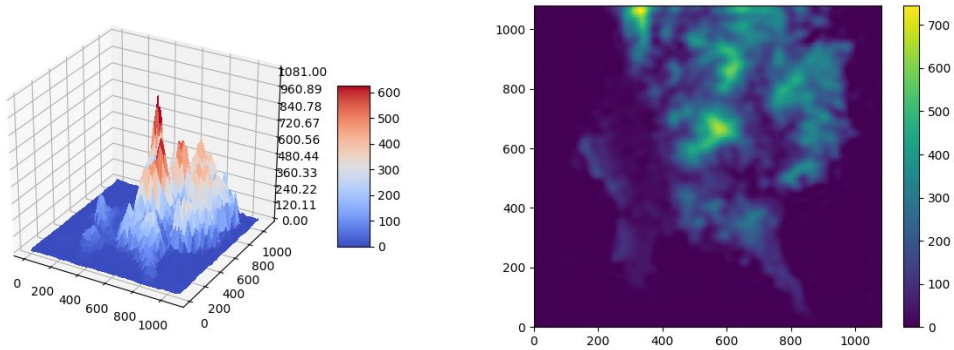
Performance

	ML	MAP	Bayesian
MSE	113.610847	135.456	**49.57**

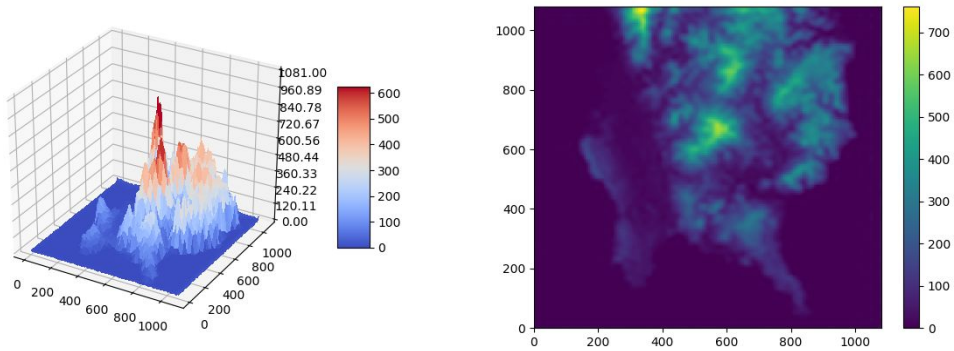
All models are trained with 4096 dimension feature vector and quantization data preprocessing technique. ML and MAP are trained with SGD and 10 epochs, 0.5 learning rate. MAP(regularization strength) is set to 0.0001 and q = 2 (Ridge). In Bayesian approach, $m0$ is set to zero, s_0 is set to $2.0I$, β is set to 25. The following figures are visualization result for these three approaches respectively.



(Visualization for Maximum Likelihood approach)

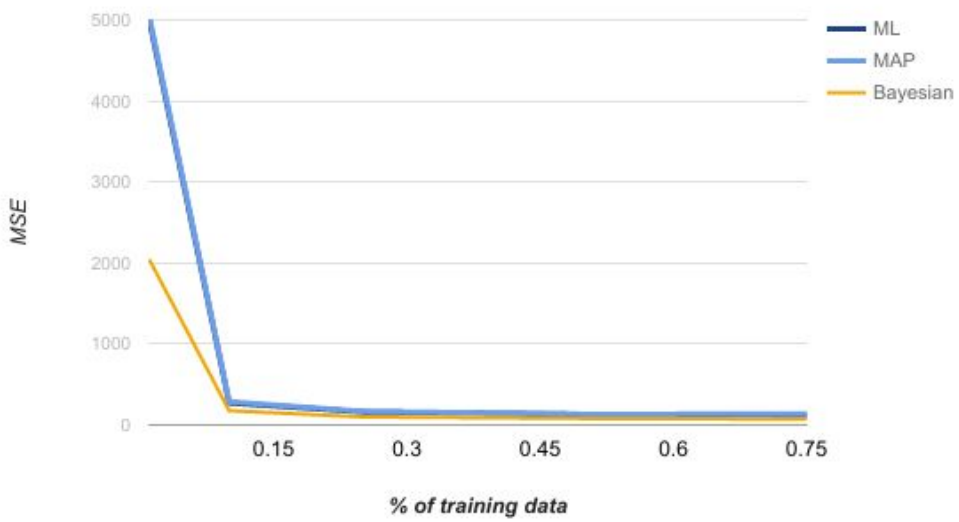


(Visualization for Maximum a posterior approach)



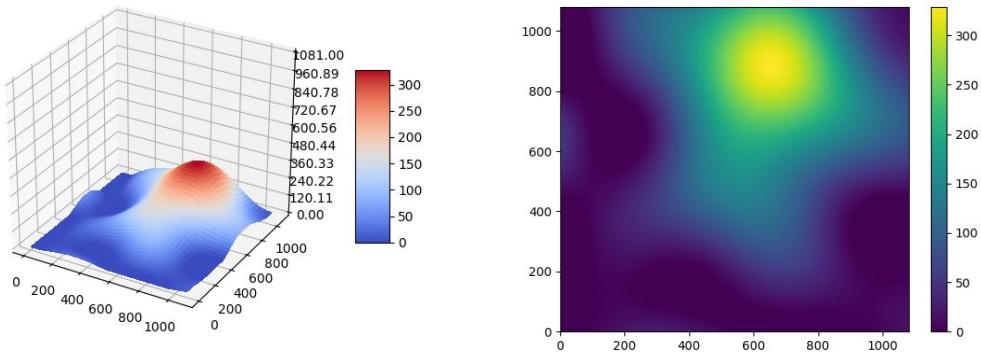
(Visualization for Bayesian approach)

To compare data efficiency between these three methods. The following figure shows performance in various size of training set of three methods. (all hyper parameters are same as the above). When we only have limited training data, we can get best performance with Bayesian approach according to Fig.6.

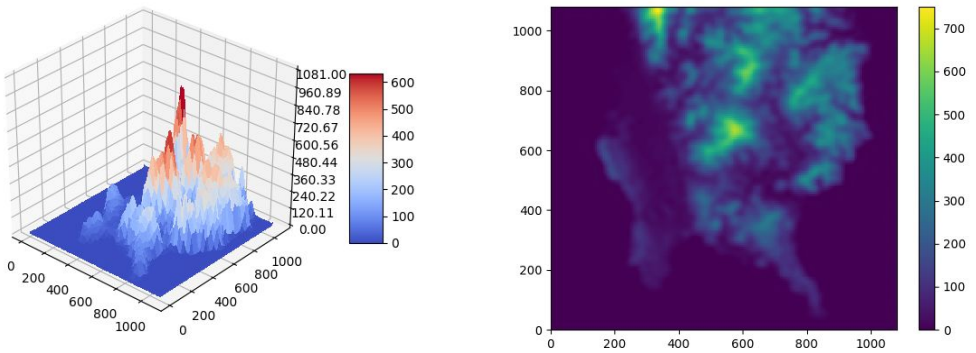


(Fig.6 Performance with different size of training data)

Under fitting and Over fitting



(Fig.7 Under-fitting (Left) 3D visualization (Right) 2D visualization)

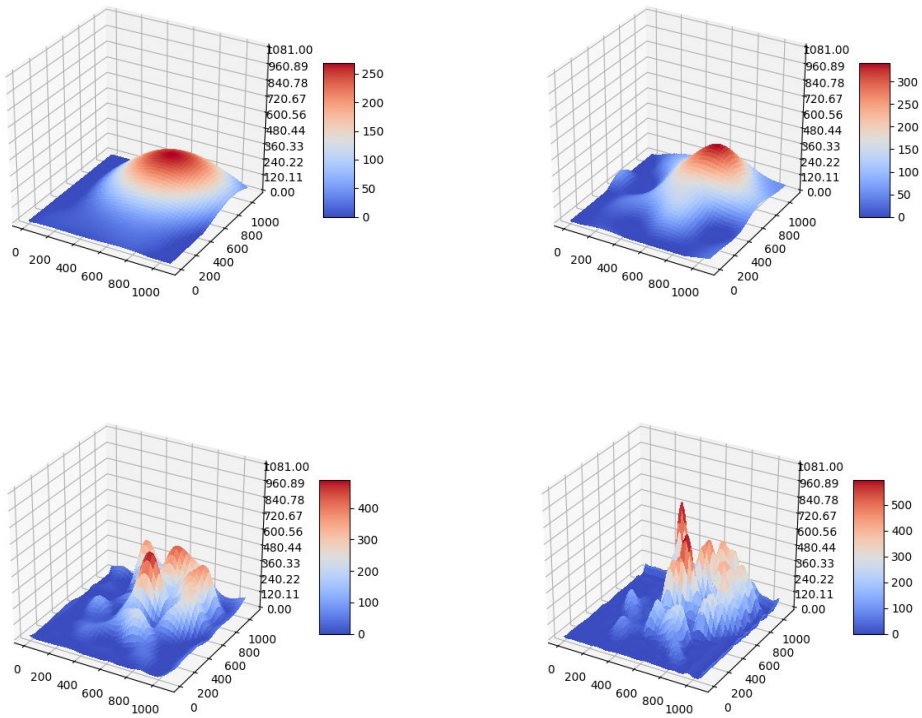


(Fig.8 Over-fitting (Left) 3D visualization (Right) 2D visualization)

I decrease dimension of feature vector to make under fitting model and vice versa. It is obvious that under-fitting model seems more smooth than over-fitting model. In other word, under-fitting model might not be able to predict terrain precisely.

Model complexity

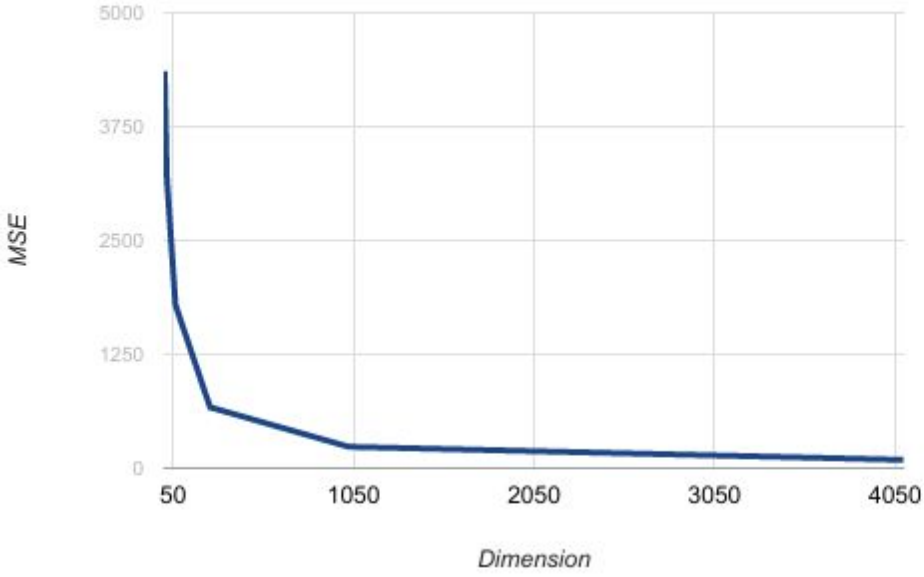
Model complexity is important. We always suffered from curse of dimensionality with too many basis function (i.e. dimension of feature vector is too high). In order to study the influence of dimension of basis function, I run experiments with different dimension with Bayesian approach. The following figures illustrates the influence of change of number of basis function with the other hyper parameters fixed. (*d* is abbreviation of dimension in the following four figures.)



(Top left: d = 9, Top right: d = 25, Bottom left: d =100, Bottom right: d = 400)

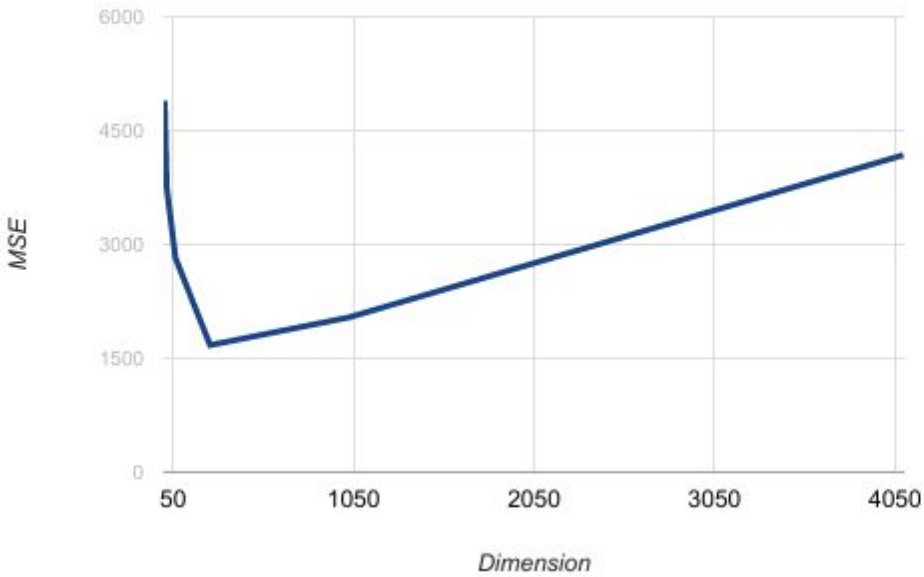
Cross validation

All experiments are carried with K = 10 (i.e. 10-fold cross validation) settings.
First, I want to study the influence of dimension of feature vector. Since I am only interested at dimension of feature vector, I choose Bayesian approach to do experiment.



(Fig.9 Result with feature vector dimension change on Bayesian)

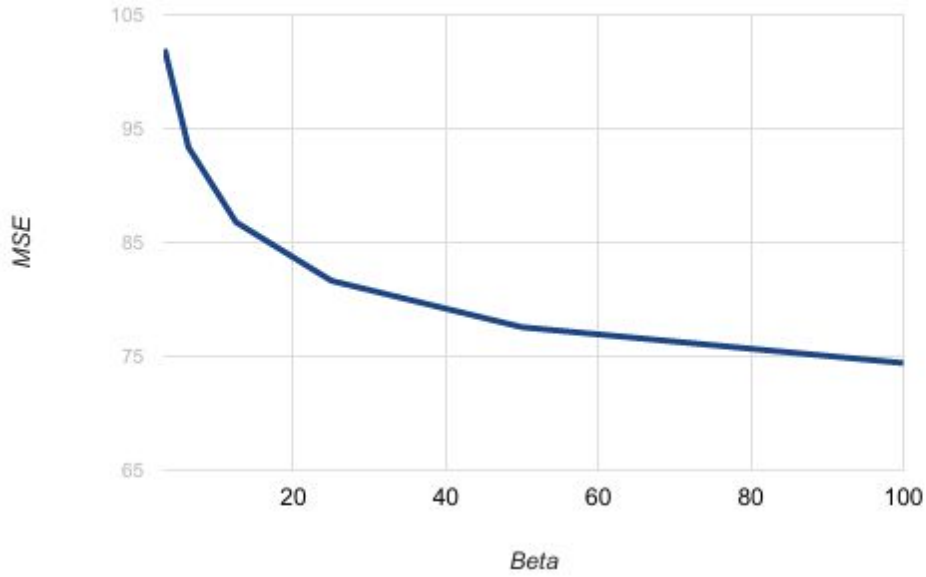
MSE decreases rapidly with increasing dimension. It makes sense that accuracy depends on model complexity (i.e. number of parameters). Fortunately, amount of training data is enough to overcome the curse of dimensionality. What if we train the model in high dimensional feature space with few training data ? The result is showed in Fig.10.



(Fig.10 Result with feature vector dimension change on Bayesian with only **1%** training data)

Fig.10 tells us that when there is only few training data, we would be easily suffered from curse of dimensionality.

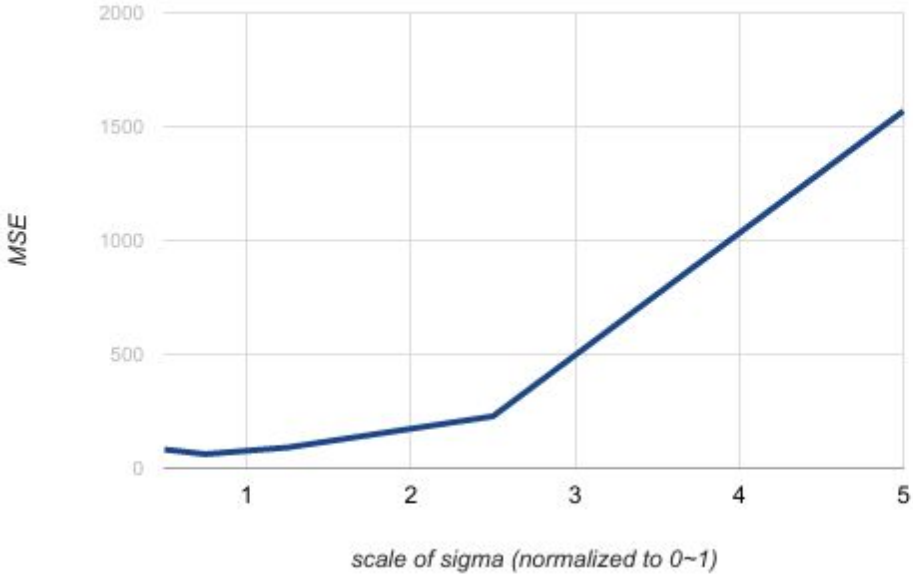
In addition, I want to show the influence of noise model on Bayesian approach.



(Fig.11 Result with β change on Bayesian)

Variance in Gaussian basis function

When we choose Gaussian function as basis, we need to pre-define the variance of each basis. It is important to choose a suitable variance for each case. If variance is too small, the model would be very sharp and generalizability can be so poor also. While too high variance might cause the model to be under-fitting.



(Fig.13 Result with variance change on Bayesian)

According to Fig.13, we can conclude that in this experiment settings, there exist an optimal value. The smaller variance doesn't always decrease MSE.