Machine Learning

Homework one

0550193 陳昱瑋 電子所 05

Due: Mar 21st, 2017

Outline

A. Data and plan

The training data has 40,000 set of X, Y and height, and Fig. 2 is the contour from x-y plane. We can find that:

1. Data are evenly distributed in region from 1 ~ 1081
2. The contour is quite obvious using training data

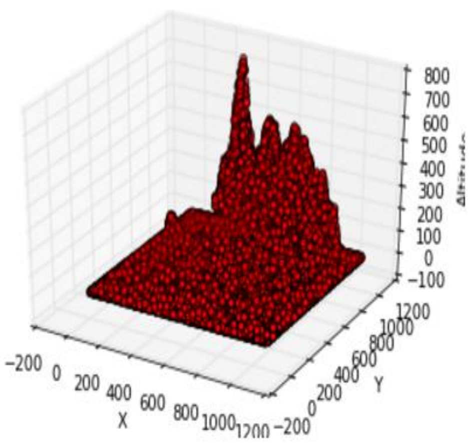


Fig. 1 plot in Python

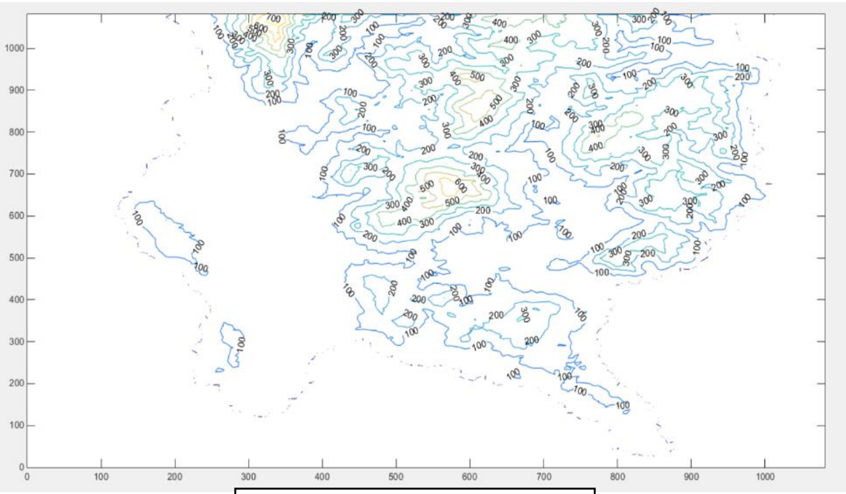
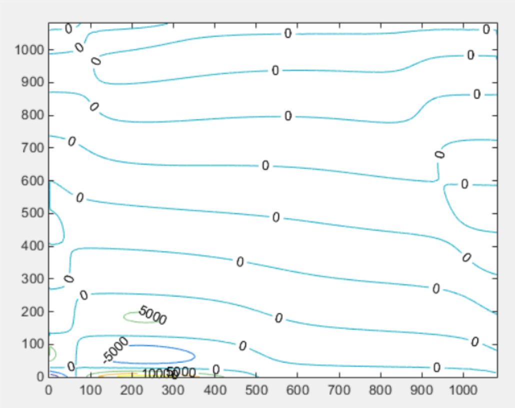
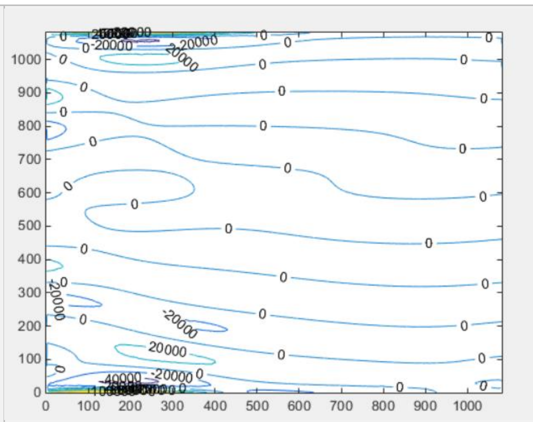
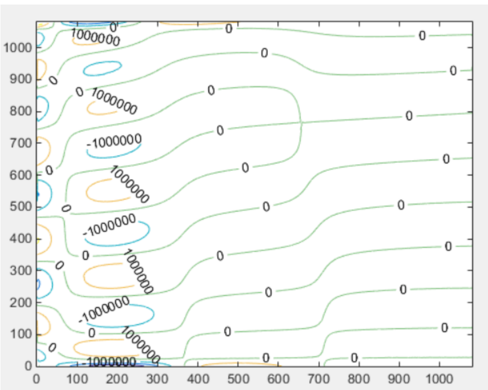
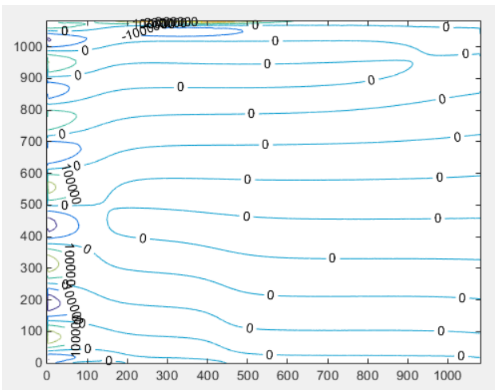


Fig. 2 plot in Matlab

Just before training the model of ML, MAP and Bayesian, I will divide training data into four sets, each with 10,000 data. The cross-validation we are using is 4-folder.

B. Model Explanation

1. From the rough model using x and y partition in different (spacing).

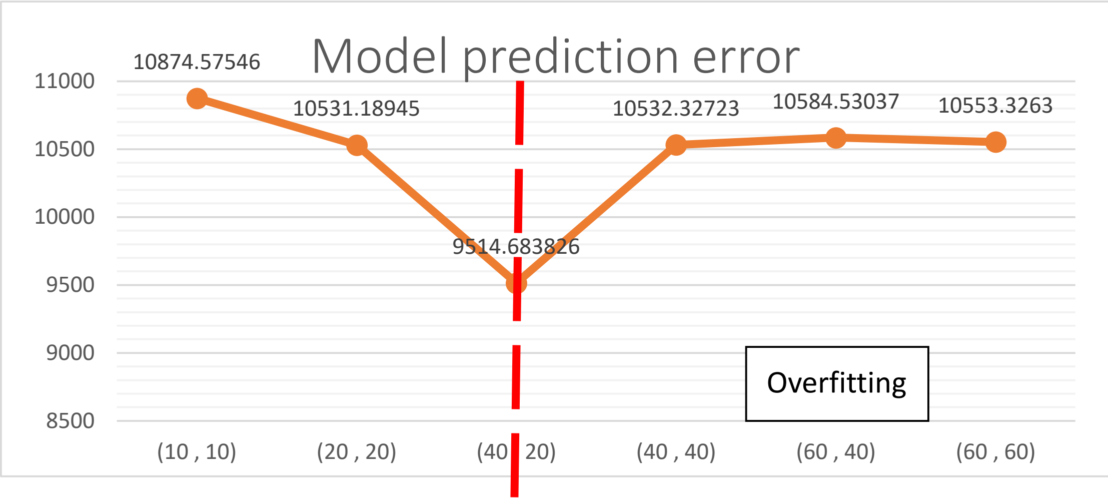
<table>
<tr><td colspan="2">Error for fixed variance:
76974212983.67911</td></tr>
<tr><td>
Tile size(x*y): 10* 10
Mean:
X) 1~1081, space 108
Y) 1~1081 space 108
Variance : fixed to 10^5.</td><td>
Tile size(x*y): 10* 20
Mean:
X) 1~1081, space 108
Y) 1~1081 space 54
Variance : fixed to 10^5.</td></tr>
<tr><td>
Tile size(x*y): 20* 10
Mean:
X) 1~1081, space 108
Y) 1~1081 space 108
Variance : fixed to 10^5.</td><td>
Tile size(x*y): 20* 20
Mean:
X) 1~1081, space 54
Y) 1~1081 space 54
Variance : fixed to 10^5.</td></tr>
</table>

I have found that the definition in x will have more impact than definition in y.

2. Feature Matrix $\boldsymbol{\phi}$: Gaussian Mixture Model

$$\boldsymbol{\phi}_n(x_i\,,\,y_i) = exp\left(-\frac{(x_i - \mu_{n_x})^2}{2s_{n_x}^2} - \frac{(y_i - \mu_{n_y})^2}{2s_{n_y}^2}\right)$$

3. The spacing interval for model prediction relationship is shown below:



In general, the smaller the spacing is the less error will produce. However, this kind of good property all comes a price. The computing time for 40*40 titles is roughly 3 minutes, 100 * 100 titles, in contrary, will take 15 minutes or even more! I will use 40*20 titles for 40,000 data training (the best in the graph) and higher definition for cross- validation.

≑ Maximum likelihood approach (ML) feature vector

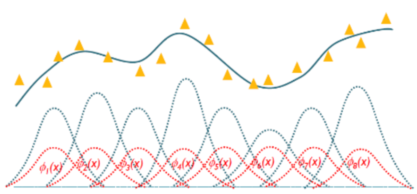
Data setting

40,000 data to train model

Likelihood

$$\mu_{n_{x,y}} = \frac{\sum_{i=1}^{M} f(x_i, y_i) x_i, y_i}{\sum_{i=1}^{M} f(x_i, y_i)} \quad , M \;\; is \;\; number \;\; of \;\; data$$

$$y = w_0 + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + \ldots + w_{M-1}\phi_{M-1}(\mathbf{x})$$



$$Var_{n_{x,y}} = \frac{\sum_{i=1}^{M} (f(x_i, y_i) - \mu_{n_{x,y}})^2}{\mathrm{M}} \quad , M \;\; is \;\; number \;\; of \;\; data$$

Solve Weight

$$\vec{\boldsymbol{w}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\vec{\boldsymbol{t}}$$

Error Estimation

$$E(\boldsymbol{w}) = \frac{1}{2K}\sum_{k}^{K}\|y(\boldsymbol{x}(k),\boldsymbol{w}) - t(k)\|^2$$

≑ Maximum a posteriori approach (MAP)

Data setting

40,000 data to train model

Likelihood

$$\mu_{n_{x,y}} = \frac{\sum_{i=1}^{M} f(x_i, y_i)x_i, y_i}{\sum_{i=1}^{M} f(x_i, y_i)} \quad ,M \ \ is \ \ number \ \ of \ \ data$$

$$Var_{n_{x,y}} = \frac{\sum_{i=1}^{M}(f(x_i, y_i) - \mu_{n_{x,y}})^2}{\mathrm{M}} \quad ,M \ \ is \ \ number \ \ of \ \ data$$

Solve Weight

$$\vec{\boldsymbol{w}} = \boldsymbol{\lambda I} + (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\vec{\boldsymbol{t}}$$

Error Estimation

Root Mean Square error (E_{QRS})

$$E_{RMS} = \sqrt{\frac{\sum_{j=1}^{NM}(y_j - t_i)^2}{NM}}$$

≑ Bayesian Estimation

When we use the Bayesian, we have assumed w is zero-mean isotropic Gaussian N (0, ß). According to the class note：

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

with $p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1})$

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

$$\Rightarrow \quad p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) = \mathcal{N}(t|\mathbf{m}_N^{\mathrm{T}}\phi(\mathbf{x}), \sigma_N^2(\mathbf{x}))$$

Because of the result,

$$p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) = \mathcal{N}(t|\mathbf{m}_N^{\mathrm{T}}\phi(\mathbf{x}), \sigma_N^2(\mathbf{x}))$$

therefore the maximum probability or average probability will be $m_N^T \Phi(x)$, and because the Gaussian prior $w_{map}=m_n$, the target t will be the same as

$$y \ = m_N^T \Phi(x). (as \ \ MAP \ \ result)$$

C. Result

for the original training dataset, 40,000 data set, and model with 11 * 11 = 121 feature vector and fixed variance (around 10 in power of 5), I have weight that is quite strange as below, the weight of each feature, it goes either too high or too low.

Fixed variable with 10 in power of 5

```
gcm.weight
array([[  5.98087111e+15],
       [ -3.87502210e+16],
       [  1.07652739e+17],
       [ -1.53769290e+17],
       [  7.56609324e+16],
       [  1.41772909e+17],
       [ -4.01986607e+17],
       [  6.48469434e+17],
       [ -9.04661756e+17],
       [  1.08001184e+18],
       [ -9.37418732e+17],
       [  4.39443597e+17],
       [  3.82767828e+16],
       [ -7.97617321e+16],
       [ -2.70032953e+17],
       [  5.86024694e+17],
       [ -5.82216592e+17],
       [  3.52606255e+17],
       [ -1.33618275e+17],
       [  2.89809577e+16],
```

Trained variable

```
gcm.weight
array([[ -2.01085158e+09],
       [  5.32645203e+09],
       [  1.92352807e+09],
       [  7.47213581e+08],
       [ -1.53954901e+09],
       [ -2.07358632e+09],
       [ -4.97074378e+08],
       [  4.44412783e+09],
       [  3.02208938e+09],
       [ -4.11129116e+09],
       [ -5.48299224e+09],
       [ -4.71032489e+09],
       [ -3.23071602e+09],
       [ -9.19358963e+08],
       [  3.19001355e+09],
       [  5.75593517e+08],
       [  5.08241576e+09],
       [ -4.42547017e+09],
       [  4.04730314e+09],
       [ -3.45737258e+09],
```

D. Discuss

1. I cannot produce the optimal (or at least acceptable) solution in time mostly due to how I pick mean and variance. Apparently, I cannot set the mean section too small due to time and memory issue I've mention before[B.3], and again, the solution will as well-optimizer as better computing ability I have.
2. Instead of using ML, MAP, maybe the pre-data will do more efficient work. The question this homework ask is intuitively solved with numerical skill (linear regression) and might have predicted less error result.
3. I neither had time nor computing for finding relationship between minimize the interval and error function. That will be the interesting task after all.

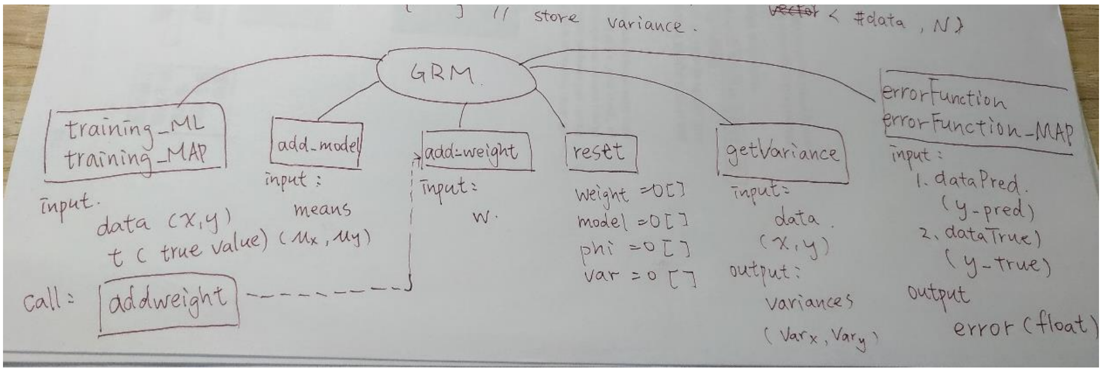
E. Reference

Mostly class note and Python document

F. Code

```python
class Gaussian_Regression_Model:

    def add_model(self,means):
    def add_weight(self, w):
    def reset(self):
    def getVariance(self, data):

    def training(self, data, t):
    def training_MAP(self, data, t, lamb):

    def errorFunction(self, dataPred, dataTrue):
    def errorFunction_MAP(self, dataPred, dataTrue):

    def prediction(self, data):
```

As I've mentioned above, I created a class of Gaussian Regression Model so that it will save my time debugging the code (as well as code maintains).



Appendix I

≑ Memory issue

I had some hard time programming in python and understanding the model constrain. This is quite exciting for me to actually train a model and learning by doing. Special thanks to my friends for advice and help me with the model.

```
phi : 40000 times 121
phi_trans : 121 times 40000
phi_inv : 121 times 121
 t : 40000 times 1

---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
<ipython-input-5-a768fb2529b6> in <module>()
      8 height = np.asarray(height).reshape(40000,1)
      9 #modeling
---> 10 gcm.modeling(position,height)

<ipython-input-2-3b65701686cf> in modeling(self, data, t)
     41         a = np.dot(self.phi,phi_inv)
     42         b = np.dot(a,phi_trans)
---> 43         weight = np.dot(b,t)
     44         print weight
     45         self.weight = weight

MemoryError:
```

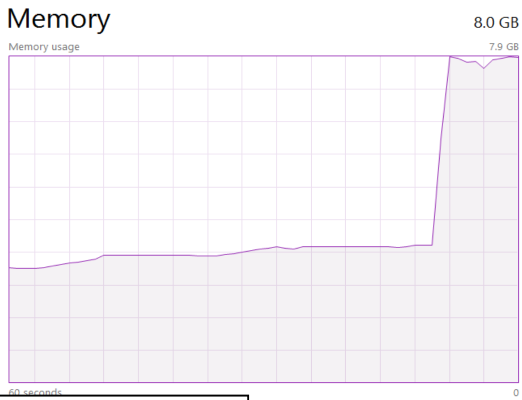


Fig. 3 memory error for using class multiple times

I have faced may problem than expected, it took me more time for debugging than coding or training; however, after understanding the problem and think it through before I mess up with everything really help me save lots of time.

≑ Python class

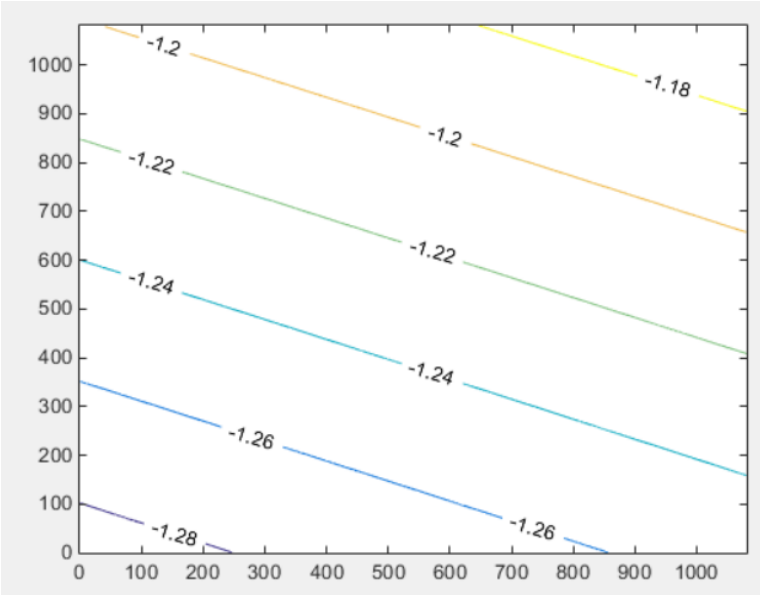
It is much easier to debug or training with different dataset. For example, it only takes me 10% of my time to train the result I want, but it takes 80% for building a model for ML, MAP and Bayesian model prediction. In general, using OOP programming skill will help for the next couple times homework and final projects.

≑ Post handout work (before demo)

This predict model is

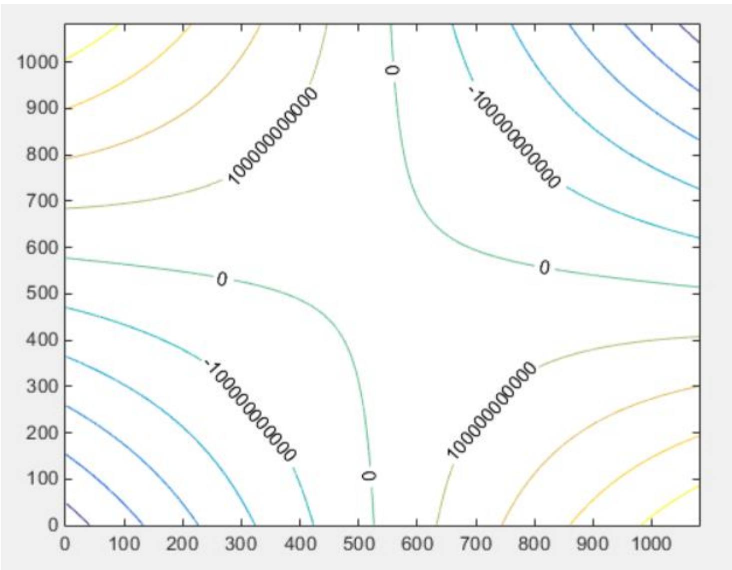
$$\hat{y} = \Phi \mathrm{w}$$

Which is not so correct but the concept is quite the same for only discussing how to cut the data.



I have award that I might use the wrong mode for predicting $\hat{y}$, and it should be as the class not write:

$$\hat{y} \ = \Phi(\Phi\Phi^T)^{-1}\mathrm{w}$$



In the end, I can't get my final result due to wrong choosing variance and means…