Machine Learning - hw3

Huiting Hong A061610

January 5, 2018

Abstract

This assignment is going to discuss 3 problems, which are Gaussian-Process-for-Regression, Support-Vector-Machine and Gaussian-Mixture-Model. This report will focus more on discussing the problem while implementation details please go to my github repository[1] to see more detail.

1 Gaussian Process for Regression

1.1 Implement the Gaussian process by using the exponential-quadratic kernel function

For the prediction result base on different theta value could be seen in Figure1

For the root-mean-square errors for both training and test sets, we can see the results as follows:

RMS		
$\theta 0$, $\theta 1$, $\theta 2$, $\theta 3$	train	test
1, 4, 0, 0	1.05224307	1.31972289
0, 0, 0, 1	6.65758954	6.63431273
1, 4, 0, 5	1.0288404	1.2842308
1, 64, 10, 0	1.03287726	1.38904624

We can found that for linear kernel the result is super bad which could also see the means curve (in red line) can't feat the data pretty well. As for the other kernel function, the RMS didn't vary a lot, which are better kernel function to describe the data.

1.2 Automatic Relevance Determination (ARD) framework

To solve this problem we have to first derive the gradient of C_N on θ_0, η, θ_2, θ_3. The derivation detail can be seen in Figure2.

In Figure 3 we can see the values of the hyperparameters as a function of iterations.

And the best theta after calculation is as follows:

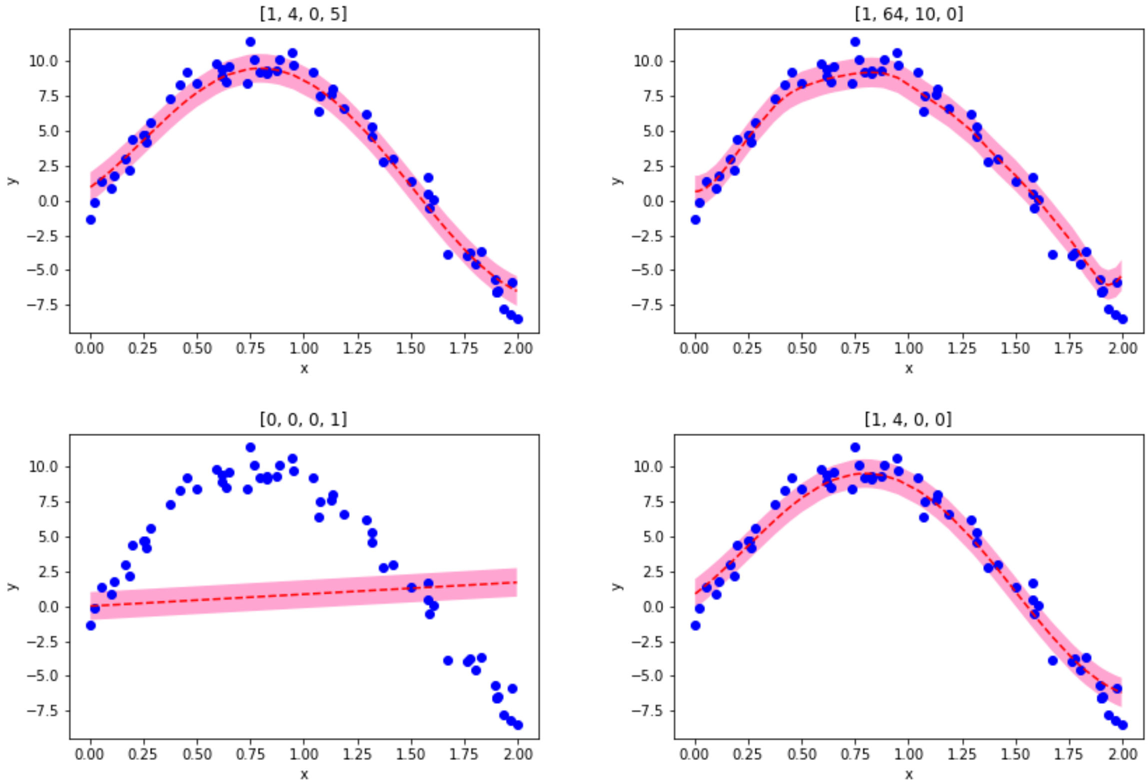

Figure 1: The four figure show the prediction result on trining set base on different hyperparameters θ, which the blue dots means the training data.

Best theta Result	
$\theta 0$, η, $\theta 2$, $\theta 3$	3.46430312, 6.02881253, 4.00613393, 5.00045099

In Figure4 we can see the prediction result base on the best theta value:
And the corresponding RMS is as follows:

RMS		
$\theta 0$, η, $\theta 2$, $\theta 3$	train	test
Best	0.83939913	1.07229143

Finally we can compare with the Bayesian Result in Figure5:

We can see the Baysian result oscillate much more and its RMS error is worse than the Gaussian result. We believe it's the reason that Gaussian have more information on modeling the noisy part(the parameter beta).

Compare RMS		
$\theta 0$, η, $\theta 2$, $\theta 3$	Baysian	G.S. Process
Best	7.36480426	0.83939913

$$\frac{\partial}{\partial \theta_0} C(x_n, x_m) = e^{-\frac{1}{2}\sum_{i=1}^{D} \eta_i (x_{ni} - x_{mi})^2}$$

$$\frac{\partial}{\partial \eta_1} C(x_n, x_m) = -\frac{1}{2}\theta_0 (x_{ni} - x_{mi})^2 \cdot e^{-\frac{1}{2}\sum_{i=1}^{D} \eta_i (x_{ni} - x_{mi})^2}$$

$$\frac{\partial}{\partial \theta_2} C(x_n, x_m) = 1$$

$$\frac{\partial}{\partial \theta_3} C(x_n, x_m) = x_n^T \cdot x_m$$

Figure 2: The derivation result base on four different parameters.

Too see more coding implementation detail please refer to my github repository[1].

2 Support Vector Machine (SVM)

This part can be divided into four part, to keep it simple I will just show the four results of SVM. (See Figure 6)

2.1 Influence of dimension reduction

We believe the reduce of dimension will improve the result.

Too see more detail please refer to my code[3]

3 Gaussian Mixture Model

3.1 Kmeans result

We can see the μ_k table as in Figure 7:

The log likelihood curve of GMM is as follows:(In Figure8)
The resulting image is shown in Figure9)
Too see more detail please refer to my code[4]

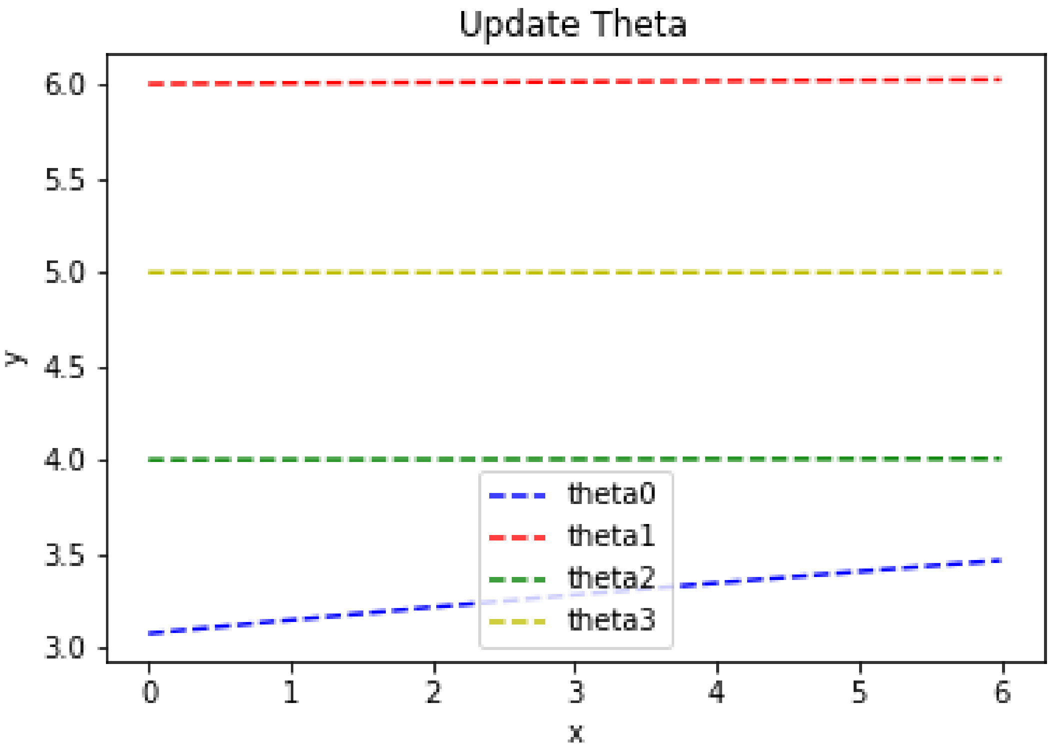

Figure 3: The update theta base on 0.01 learning rate.

References

[1] https://github.com/w102060018w/2017_NCTU_MachineLearning_hw3.

[2] https://github.com/w102060018w/2017_NCTU_MachineLearning_hw3/blob/master/Problem1_

[3] https://github.com/w102060018w/2017_NCTU_MachineLearning_hw3/blob/master/problem2.

[4] https://github.com/w102060018w/2017_NCTU_MachineLearning_hw3/blob/master/Problem3_

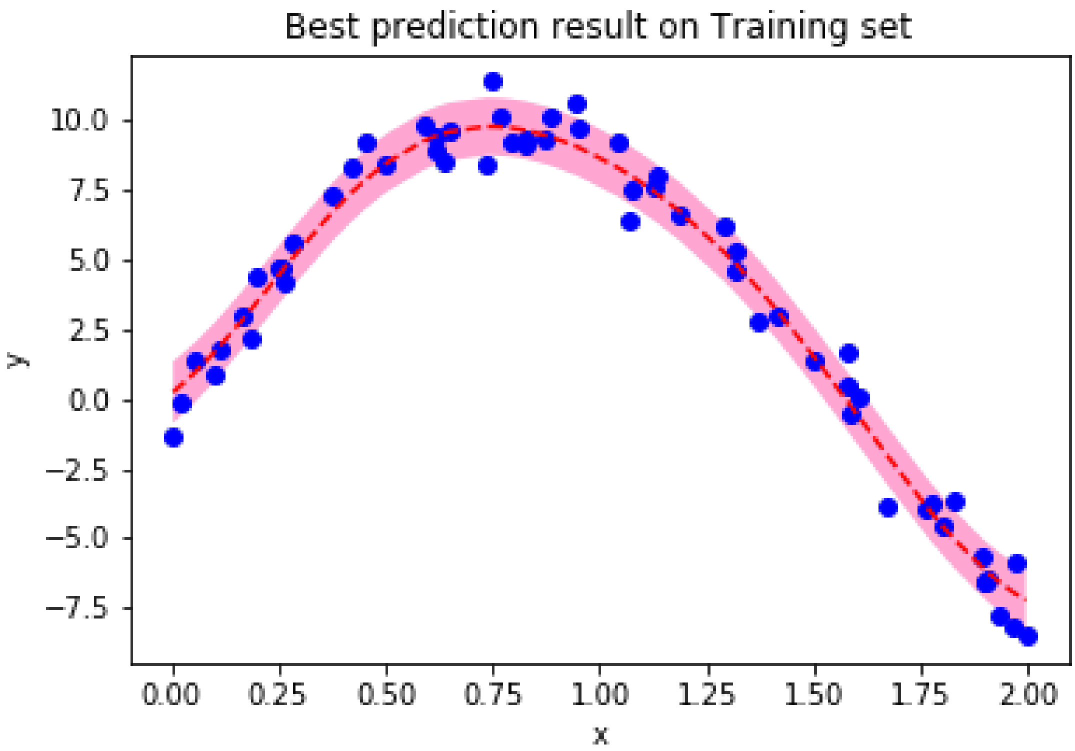

Figure 4: The update theta base on 0.01 learning rate.

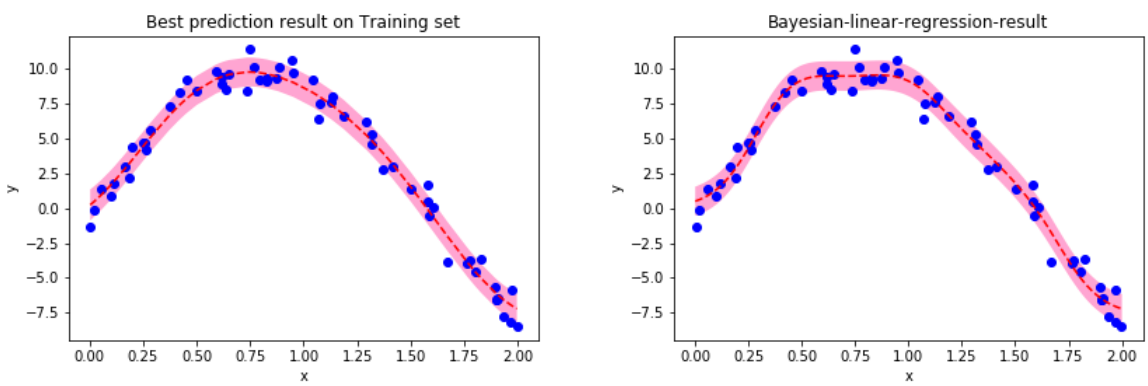

Figure 5: On the left hand side is the prediction result base on best hyperparameters, while on the right hand side is the prediction result base on Baysian Linear regression.

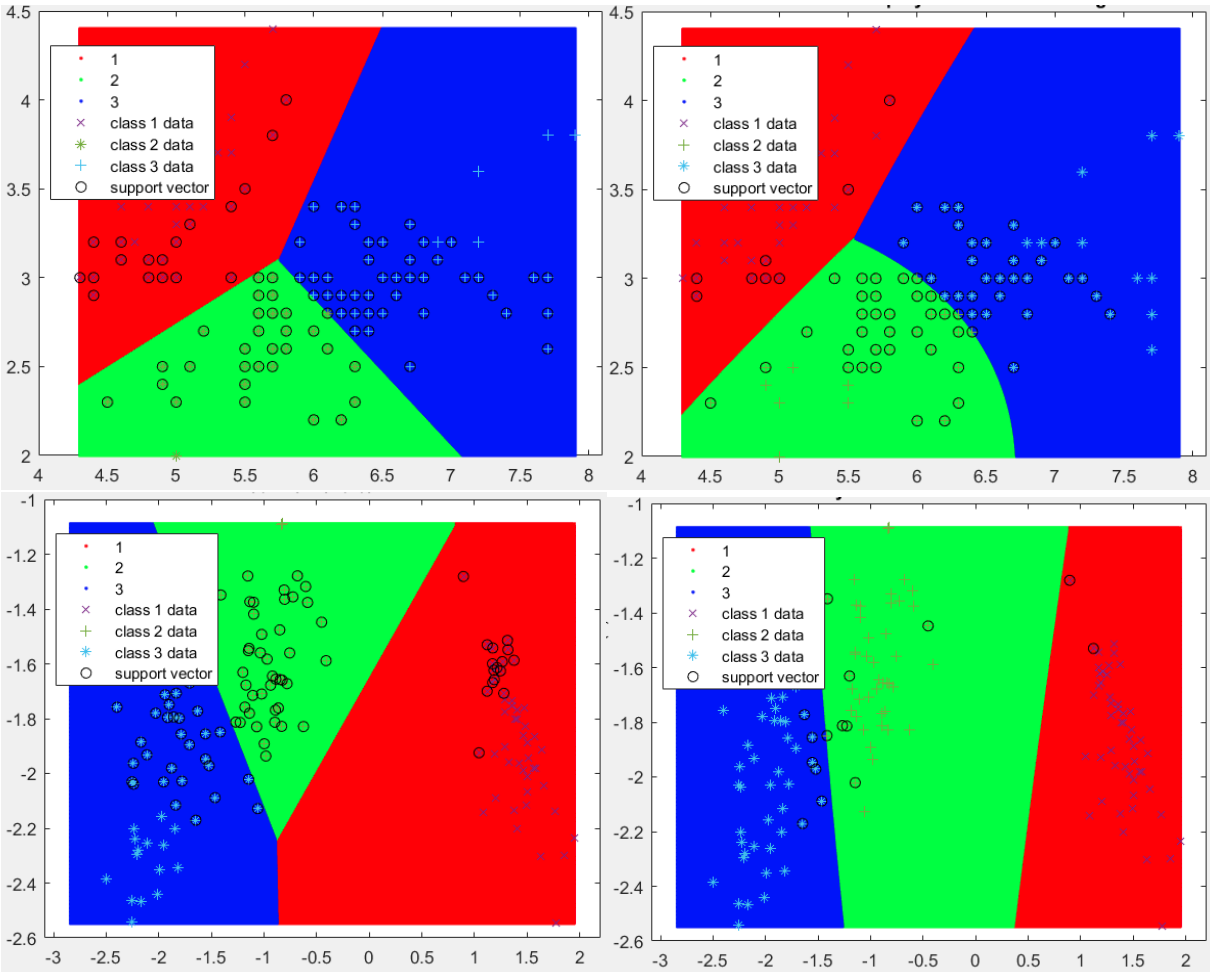

Figure 6: The Result of SVM are as shown in above. On the left-top is the first 2 feature with linear kernel. On the right-top is the first 2 feature with ploynomial kernel with degree2, on the left-bottom is the Linear kernel after LDA, and on the right-bottom is the Polynomial kernel after LDA

```
=====================================
kmeans - means =
 [[  79.14459633   46.46732166   12.93720632]
 [ 205.47309219  185.14137627  127.10013233]]
kmeans - cov =
 [[[ 4114.22834436  1425.91170936   267.15089947]
  [ 1425.91170936  1289.76665959   447.90717483]
  [  267.15089947   447.90717483   447.38990574]]

 [[ 1418.14296807   718.32835199  -110.80431933]
  [  718.32835199  1376.0900701   1290.77344843]
  [ -110.80431933  1290.77344843  4406.7725055 ]]]
kmeans - pi =
 [ 0.50802951  0.49197049]
```

```
=====================================
kmeans - means =
 [[  49.89148391   33.17430931    8.83309598]
 [ 188.98232984  133.66492147   46.32918848]
 [ 207.42582106  200.84409211  175.25443564]]
kmeans - cov =
 [[[ 1300.61568082   705.91267783   162.8773596 ]
  [  705.91267783   743.46890422   237.09944325]
  [  162.8773596    237.09944325   282.92799608]]

 [[ 2040.12730557   599.56083304  -433.94837167]
  [  599.56083304  2536.15801033   628.31809195]
  [ -433.94837167   628.31809195  1097.38312709]]

 [[ 1428.44381798   738.719088      39.26877939]
  [  738.719088     800.37395623   569.35865682]
  [   39.26877939   569.35865682  1344.23801826]]]
kmeans - pi =
 [ 0.38096788  0.33159722  0.2874349 ]
```

```
=====================================
kmeans - means =
 [[  32.51799687   20.98943662    4.48200313]
 [ 114.30901288   79.08154506   23.84855917]
 [ 150.20337302  146.85416667  116.5734127 ]
 [ 221.01267218  146.58181818   37.97134986]
 [ 217.62148685  208.65140526  181.35403445]]
kmeans - cov =
 [[[  5.14578315e+02   2.66937397e+02   2.70574444e+01]
  [  2.66937397e+02   2.54052705e+02   5.93079085e+01]
  [  2.70574444e+01   5.93079085e+01   9.45869218e+01]]

 [[  9.54704021e+02   3.00239196e+00  -6.26300882e+01]
  [  3.00239196e+00   9.06933265e+02   2.59554348e+02]
  [ -6.26300882e+01   2.59554348e+02   4.62788224e+02]]

 [[  1.73801519e+03   7.15423508e+02  -7.57108680e+02]
  [  7.15423508e+02   5.30509487e+02   2.88008846e+01]
  [ -7.57108680e+02   2.88008846e+01   1.45499263e+03]]

 [[  5.03929867e+02   1.11303371e+02  -6.19214000e+01]
  [  1.11303371e+02   3.32105212e+03   8.11396283e+02]
  [ -6.19214000e+01   8.11396283e+02   7.01717086e+02]]

 [[  4.60720282e+02   1.50311842e+02   4.34114319e+01]
  [  1.50311842e+02   4.57043940e+02   5.57495581e+02]
  [  4.34114319e+01   5.57495581e+02   1.14157412e+03]]]
kmeans - pi =
 [ 0.27734375  0.17697483  0.109375    0.1969401   0.23936632]
```

```
=====================================
kmeans - means =
 [[   3.77929688    2.52929688    2.33398438]
 [  18.24208566   10.33891993    1.88268156]
 [ 125.48615385  113.37846154   73.96923077]
 [  47.49382716  111.72839506  206.12345679]
 [  30.18112245   17.89540816    3.18112245]
 [  40.55240175   31.81222707    6.39737991]
 [  66.1875       11.203125      2.078125  ]
 [ 153.25221239  154.09070796  123.6039823 ]
 [ 122.21774194   21.24193548    4.33870968]
 [  56.87557604   42.99078341    5.5921659 ]
 [ 171.00591716   50.47928994   12.01775148]
 [  75.09         69.27          52.02      ]
 [ 164.70673077  134.94951923   45.11778846]
 [ 226.22417154   64.35867446    9.15789474]
 [  85.54830287   47.74412533    9.15926893]
 [ 107.1616       96.392         17.9184    ]
 [ 210.41964966  191.13099772  154.12338157]
 [ 214.38009788  162.31484502   70.51223491]
 [ 234.23927393  203.35643564   37.42739274]
 [ 224.40940941  227.47147147  212.09209209]]
```

Figure 7: The corresponding mu, covariance matrix and pi to different K value. On the left top is K=2, right top K=3, left bottom K=5, and right bottom K=20(Since when K=20, the size is too large to do the screen shot, I simply show the result of means.)

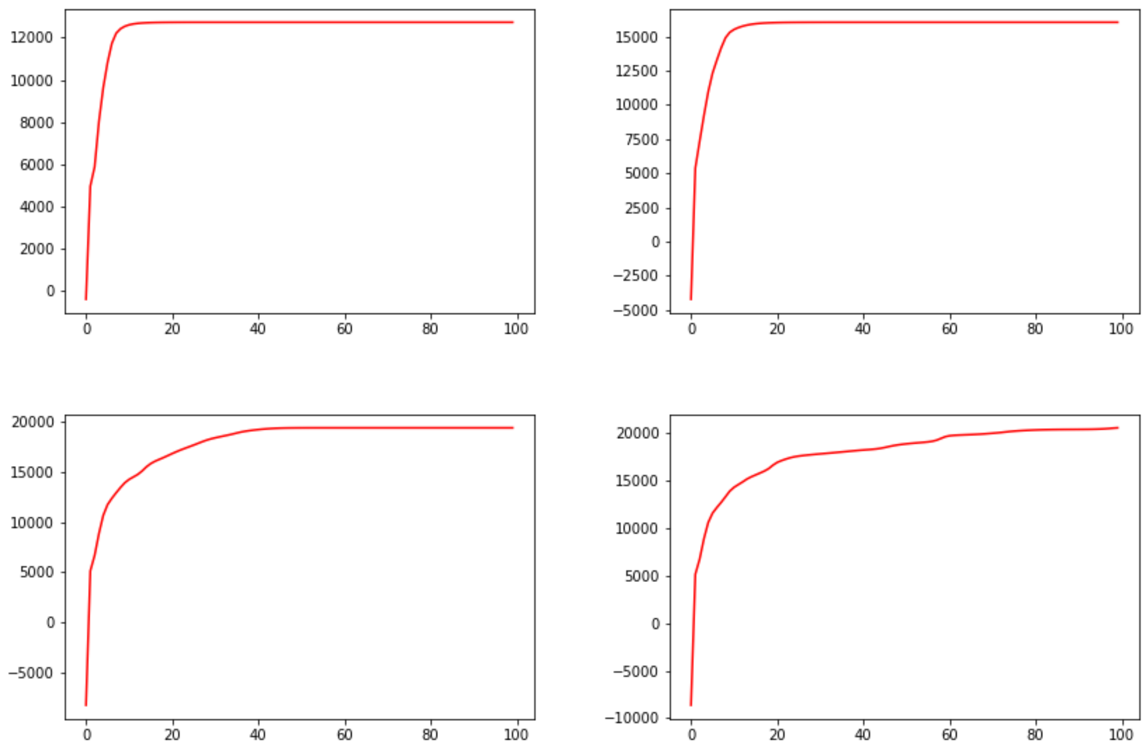

Figure 8: The log likelihood curve for different K value (when k goes to 20, it really takes lots of time though...))

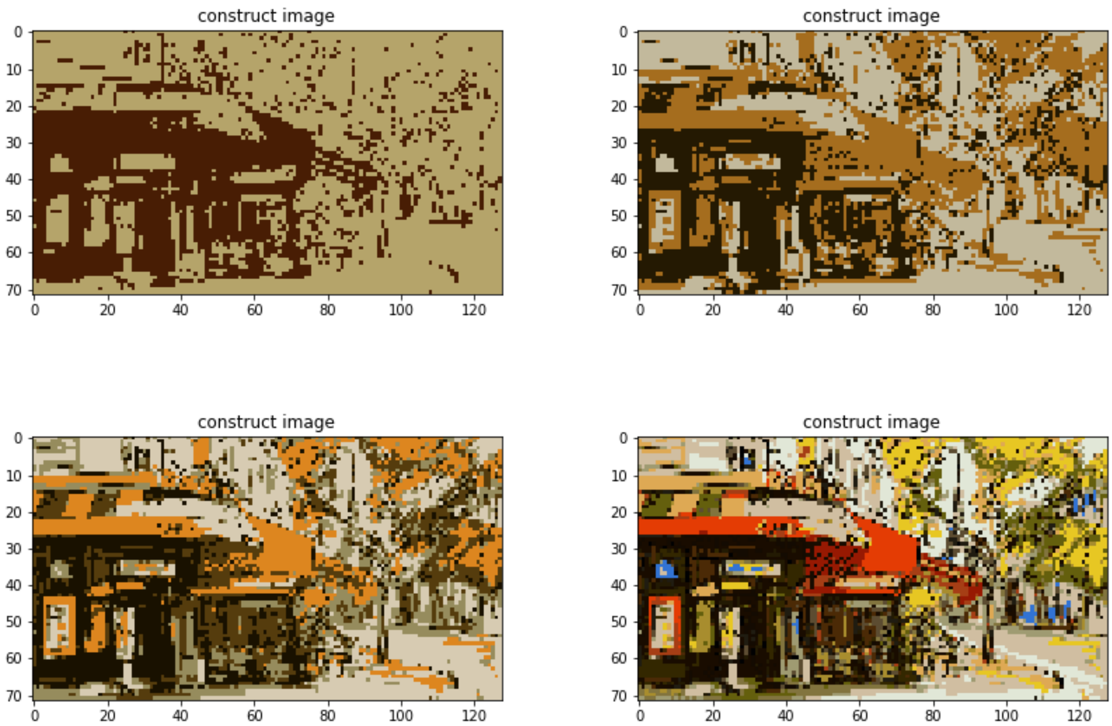

Figure 9: We can the reconstruct image in different K values.